electronics

MDPI

Article

Improved RTT Fairness of BBR Congestion Control Algorithm Based on Adaptive Congestion Window

Wansu Pan [1,2], Haibo Tan [2], Xiru Li [2] and Xiaofeng Li [1,2,*]

[1] Hefei Institute of Physical Science, Chinese Academy of Sciences, Hefei 230031, China; wansupan@mail.ustc.edu.cn
[2] University of Science and Technology of China, Hefei 230026, China; hbtan@hfcas.ac.cn (H.T.); xrli@hfcas.ac.cn (X.L.)
* Correspondence: xfli@hfcas.ac.cn; Tel.: +86-551-65591292

Abstract: To alleviate the lower performance of Transmission Control Protocol (TCP) congestion control over complex network, especially the high latency and packet loss scenario, Google proposed the Bottleneck Bandwidth and Round-trip propagation time (BBR) congestion control algorithm. In contrast with other TCP congestion control algorithms, BBR adjusted transfer data by maximizing delivery rate and minimizing delay. However, some evaluation experiments have shown that the persistent queues formation and retransmissions in the bottleneck can lead to serious fairness issues between BBR flows with different round-trip times (RTTs). They pointed out that small RTT differences cause unfairness in the throughput of BBR flows and flows with longer RTT can obtain higher bandwidth when competing with the shorter RTT flows. In order to solve this fairness problem, an adaptive congestion window of BBR is proposed, which adjusts the congestion window gain of each BBR flow in network load. The proposed algorithms alleviate the RTT fairness issue by controlling the upper limit of congestion window according to the delivery rate and queue status. In the Network Simulator 3 (NS3) simulation experiment, it shows that the adaptive congestion window of BBR (BBR-ACW) congestion control algorithm improves the fairness by more than 50% and reduces the queuing delay by 54%, compared with that of the original BBR in different buffer sizes.

Keywords: TCP congestion control; BBR; RTT fairness; congestion window

Citation: Pan, W.; Tan, H.; Li, X.; Li, X. Improved RTT Fairness of BBR Congestion Control Algorithm Based on Adaptive Congestion Window. *Electronics* **2021**, *10*, 615. https://doi.org/10.3390/electronics10050615

Academic Editor: Christos J. Bouras

Received: 24 January 2021
Accepted: 2 March 2021
Published: 6 March 2021

1. Introduction

After the observation of Internet congestion collapse in 1986, Jocobson [1] proposed a method to adjust the transmission window of TCP flow by using the loss of packet. With the development of Internet infrastructure, disadvantages of these loss-based congestion control (loss-based CC) algorithms [2–4] are gradually revealed. Once packet loss occurs, the traditional loss-based CC algorithms reduce the congestion window (CWND) and then take a long time to recover. Moreover, the queuing delay and packet retransmissions resulted in throughput degradation. Thus, new congestion control algorithms need to be proposed to adapt maximum transmission speed in the current Internet.

In 2016, a congestion control algorithm based on Bottleneck Bandwidth and Round-trip propagation time (BBR) [5,6] was proposed by Google. The algorithm adjusts its sending rate according to the estimated bottleneck bandwidth (BtlBw) and Round-trip propagation time (RTprop) in order to achieve the maximum delivery rate and minimum delay. Different from the loss-based algorithm, the goal of BBR is to operate at Kleinrock's optimal operating point [7], at which the total data inflight is equal to one Bandwidth Delay Product (BDP). The related experimental results showed that BBR can significantly improve the throughput of TCP connection, compared with traditional CUBIC [8]. Some scholars have made some modifications to the control parameters in BBR to obtain better performance [9–11] and apply it to different network fields [12–14]. Moreover, Google is actively upgrading BBR v2 by modifying some parts of the BBR algorithm [15,16].

Since BBR is still under development, many issues have been reported when BBR operates in different scenarios [17–24], including the fairness between the BBR algorithm and the other algorithm [25–28] and the Intra-Protocol Round-trip time (RTT) fairness problem of BBR [18,29–32]. One especially obvious problem is that the RTT fairness of different BBR flows can lead to throughput imbalance. Ma et al. [18] and Kim et al. [30] pointed out that, when multiple flows with different RTTs share a link, the longer RTT flows are more favored by BBR algorithm. When the bottleneck buffer is small, the longer RTT flow can achieve higher throughput and trigger higher sending rate than shorter RTT flows. Furthermore, the shorter RTT flows will be starved when flows suffer severe packet loss and high retransmission. Song et al. [28] pointed out the problem of BBR when analyzing the Inter-Protocol Fairness of BBR: BBR only estimates the delivery rate based on BDP and uses a fixed congestion window, without considering the impact of packet loss. It can be seen from the above literature that if the amount of data sent by the BBR exceeds the capacity of the link, it can lead to serious queuing delay and severe network congestion. Therefore, applying the window control method to the BBR algorithm to find the optimal node can alleviate the unfairness between different RTT flows and improve the throughput of short RTT flows. This paper proposes an improved BBR algorithm based on adaptive congestion window method.

The rest of this article is organized as follows: Section 2 introduces some related work on improving RTT fairness in BBR algorithm. The BBR algorithm is elaborated in Section 3. Section 4 expounds the RTT fairness issue and introduces the adaptive congestion window of BBR (BBR-ACW) algorithm details with a model. Section 5 evaluates the Network Simulator 3 (NS3) experimental results of the BBR-ACW by comparing with BBR and BBQ. Conclusions and discussions are carried out in Section 6.

2. Related Work

Some scholars have evaluated and improved RTT fairness in BBR congestion control algorithm. When Ma et al. [18] deployed BBR algorithms on the Internet, they found consistent bandwidth differences between competing flows, with long BBR flows inevitably consuming more bandwidth than short BBR flows. A theoretical model to characterize the dynamics of the BBR bandwidth is established by Tao et al. [29]. It is also proved that the occupation of bandwidth between different RTT flows is mainly affected by the RTT ratio. In this model, when the RTT ratio between the two flows is greater than 2, serious unfairness may occur. Zhang et al. [31] evaluated BBR and its variants on NS3, including RTT fairness, Inter-Protocol Fairness, packet loss rate, channel utilization, etc. The experimental results show that the BBR algorithm has serious bandwidth unfairness for different RTT flows.

To solve the fairness problem in the BBR algorithm, Ma et al. [18] proposed a BBQ algorithm to reduce the bandwidth consumption of the long RTT flows by limiting the detection period of long RTT flows. Yang et al. [32] proposed the Adaptive-BBR algorithm, which used the bottleneck buffer information in the BBR framework to adjust the flows sending rate and no longer used the fixed 8 cycles of *pacing_gain* in the BBR ProbeBW stage. Sun et al. [33] proposed an RFBBR algorithm to improve the fairness of BBR flows with different RTTs, and it is applied in the wireless network spindle topology scenarios. According to Kim et al. [34], the inflight cap in the BBR ProbeBW state was restricted to less than 2 BDP during the creation of queues. If the bottleneck link is not fully utilized, the behavior of the modified BBR is close to that of the original BBR; otherwise, the inflight cap was limited to 1 BDP. Then, Kim et al. [35] proposed the Enhanced BBR(BBR-E) algorithm to improve RTT fairness and maintain better throughput. In a subsequent paper [30], a Delay-Aware BBR (DA-BBR) algorithm was proposed to alleviate the RTT fairness problem, which reduces the BDP according to the RTT. The above algorithms have mitigated the RTT fairness problem to varying degrees. Obviously, in order to further improve the RTT fairness, it is still necessary to study new improved algorithms to increase the performance and practicability of the BBR algorithm.

For detailed study, we generated simple network connection in NS3 and tested optimization algorithm of BBR according to BBR implementation [36]. Through analysis of influence factors, when too much data is poured in the process of bandwidth detection, a persistent queue that is not fully released in the exhaustion time is generated. The same bottleneck queue will be shared by all BBR flows, but the queuing delay will exceed the BBR flow with smaller RTT first. With the increase of RTprop, the shorter RTT flow enters CWND-bounded mode, so the longer RTT flow can occupy more bandwidth than the shorter RTT flow. With the increase of RTT difference between different BBR flows, RTT fairness will deteriorate. Thus, we can reduce the excessive data transmission of long RTT flow by adjusting CWND, so the RTT fairness of BBR algorithm will be improved.

From what has been discussed above, a new method based on adaptive congestion window of BBR (BBR-ACW) is proposed in this paper. BBR-ACW algorithm adjusts the *cwnd_gain* of different RTT flows, so that the data of different BBR flows in the bottleneck queue can keep approximately the same.

The BBR-ACW algorithm is implemented on the basis of BBR and evaluated by NS3 in the simulated network environment. BBR-ACW has been compared with BBR algorithm, BBQ algorithm and DA-BBR algorithm when different RTT flows in different buffer sizes or different bottleneck bandwidths. The emulation results indicate that BBR-ACW algorithm has the better performance among the above algorithms. Compared with the original BBR, it can increase the Jain's fairness index [37] by at least 1.5 times and reduce the queuing delay by up to 54%.

The main contributions of this paper are as follows:

- The fairness issue of BBR has been further analyzed and revealed the mismatch between sending rate of different RTT flows and BtlBw. The impact of buffer size on the BBR-RTT fairness is also discussed.
- BBR-ACW is proposed to solve the BBR fairness issue. The RTT fairness, retransmission and delay performance under different buffer sizes are improved by adaptively adjusting CWND based on delivery rate and queue status in network congestion.
- The extensive experiments have been conducted on the NS3 platform to test the performance of the BBR-ACW. Different bottleneck bandwidth or buffer size are configured in the experiment. From the experimental results, it can be seen that BBR-ACW improves the fairness and reduces the delay, which is better than BBR and BBQ. In addition, compared with the latest DA-BBR congestion algorithm, the fairness of BBR-ACW is slightly better, and the latency is greatly reduced.

3. BBR Overview

In the Section 3, a detailed description of the operating points of TCP congestion control is given. A brief introduction about four states of the BBR algorithm are also given.

3.1. Algorithm Principle of BBR

TCP congestion control aims to maximize the use of bottleneck link bandwidth in the network, which determines the maximum transmission rate. The amount of data inflight matches the BDP and determines the maximum delivery rate and the minimum delay simultaneously, shown in Figure 1. The loss-based CC algorithm gradually increases the CWND by slow detection and quickly reduces the CWND once the packet loss is found. When the buffer overflows, a large number of packet losses will occur, and point B is exactly where the packet loss occurs. This way can guarantee the maximum bandwidth but not the minimum transmission delay.

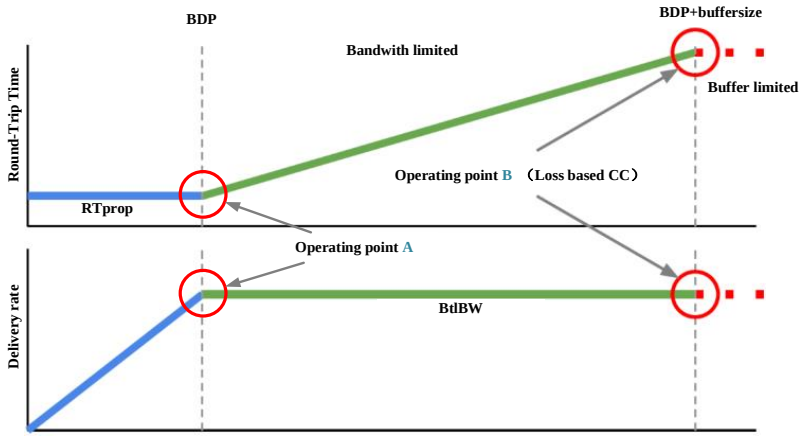

Figure 1. Delivery rate based on the amount of data inflight.

Compared with the operating point based on the loss-based CC algorithm, the point on the left edge of the bandwidth limitation area achieves the lower delay throughout the process. The optimum operating point is the operating point A in Figure 1 and is proved by Leonard Kleinrock [7]. The optimal throughput and transmission delay can be obtained at point A, whereas the distributed algorithm cannot reach the Kleinrock's point, which is proved in [38].

BBR solves the problem of finding the optimal operating point (point A) by alternately measuring the maximum delivery rate and minimum RTT in the bottleneck link. In the BBR algorithm, the TCP link can be regarded as a "pipe." The "pipe" diameter is BtlBw, the length of this "pipe" is RTprop and the volume of the "pipe" is BDP. When the amount of data in the "pipe" is BDP, this means that the sending window is equal to BDP. Then, the throughput can reach the maximum, that is, working at Kleinrock's optimum operating point (operating point A). BBR sets the maximum bandwidth (delivery rate) in the recent 10 round-trips as the current BtlBw and obtains current RTprop by detecting the minimum RTT measurement (*RTTmin*) in the past 10 s.

BBR controls its sending behavior by pacing rate and CWND: the pacing rate (sending rate) is adjusted by different *pacing_gain* to explore more bandwidth, and the CWND is set to a small multiple of BDP to limit the amount of inflight data.

When the inflight data is smaller than 1BDP, the RTT will be a fixed value, because the bottleneck link is not queuing and the buffer is not occupied at this time, and the throughput will increase with the increase of the sending rate. When the inflight data exceeds the 1BDP, the buffer will start to be occupied, and then the RTT will increase with sending rate, but the throughput will not increase after reaching the maximum value (2BDP). Once the buffer is full, the excess data packets will be discarded. The BBR algorithm always maintains the sending rate near the Operating point A, which can ensure higher throughput (filling up the pipeline) and lower transmission delay.

3.2. Four States of BBR

There are four control states in the BBR congestion control algorithm: StartUp, Drain, ProbeBW and ProbeRTT (Figure 2).

The StartUp is like the slow start in TCP. The sender can detect the maximum available bandwidth by making *pacing_gain* = 2/ln2 (approximately 2.85) to increase its sending rate. The redundant queues about 2BDP can be created at this stage. When the newly estimated bandwidth has not exceeded 1.25 times of the last maximum bandwidth for three consecutive times, the algorithm considers that the link is completely filled, and then the state turns Drain. Drain's *pacing_gain* changes to ln2/2 to decrease the sending rate until the packets in the link match the BDP, and then state changes from Drain to ProbeBW. After the StartUp and Drain, the BBR obtains the maximum delivery rate and the *RTTmin*.

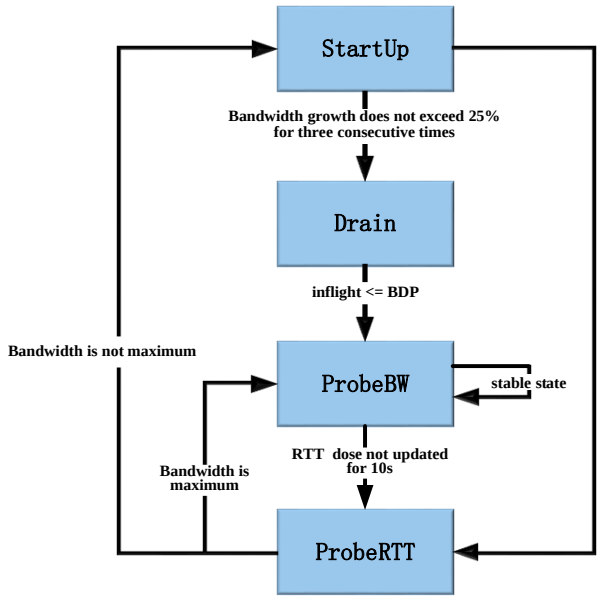

Figure 2. BBR operational state.

During the ProbeBW, the *pacing_gain* in eight cycles with different values $pacing\_gain = [1.25;\ 0.75;\ 1;\ 1;\ 1;\ 1;\ 1;\ 1]$, and each cycle lasts for *RTTmin*. The probe up phase with *pacing_gain* of 1.25 is to increase the sending rate to probe more available bandwidth, and the probe down phase with *pacing_gain* of 0.75 is to eliminate the excess queues, which accumulated in the previous probing. Then, the *pacing_gain* is 1 for the remaining six cycles to stabilize the sending rate. Furthermore, the *cwnd_gain* in the ProbeBW is set to a fixed value of 2, so that enough data packets can be sent during the probe up phase. If new *RTTmin* is not sampled again within 10 s, the link will be considered falling into congested and enters the ProbeRTT. In ProbeRTT, CWND is set to a very small value ($4 \times$ MSS) to remove all possible queues, and this state lasts for 200 ms to ensure the new value *RTTmin* to be sampled. After this process, the transition to ProbeBW or StartUp depends on whether the network bandwidth is full.

4. BBR-ACW: Adaptive Congestion Window of BBR

Here, we discuss the RTT fairness of BBR and analyze the reasons why the long RTT flows has the advantage. BBR-ACW is proposed to improve RTT fairness and reduce latency by adjusting the *cwnd_gain* according to the delivery rate and RTT of BBR flows.

4.1. Analysis of BBR's RTT Fairness

When BBR flows with different RTT flows traversing through the bottleneck link, BBR flows will obtain the maximum delivery rate and *RTTmin* after StartUp and Drain. In the ProbeBW stage, BBR will periodically deliver more flow to the bottleneck link at a rate of *pacing_gain* = 1.25; the delivery rate increases with the increase of inflight, resulting in the total sending rate being greater than the bottleneck bandwidth and forming a persistent queue on the bottleneck. This process is looped until some flows are restricted by CWND and their sending rates are reduced. Even if some flows in the probe up process are terminated, the conclusion above is still true, because other flows will quickly preempt the spare bandwidth. Once a persistent queue is formed at the bottleneck, the throughput of the flow is determined by its queue share, which eventually leads to RTT fairness issue.

On the one hand, when the BBR detects the larger bandwidth, the estimated BDP value of the shorter RTT flow will be smaller than that of the longer RTT flow. Therefore, the shorter RTT flow injects less packet into pipe, which will reduce delivery rate. The continuous low delivery rate will lead to a smaller value to be obtained in the next bandwidth

measurement, and then to a lower sending rate again, which will further reduce its queue sharing and eventually lead to a serious decrease in bandwidth.

On the other hand, the *cwnd_gain* of BBR is set to 2. When BBR flows run on a link, sending excessive packets results in the generation of queues and the increase of measured RTT. When the buffer is not enough, the queuing delay will increase with queue development. Once the queuing delay is greater than *RTTmin* (inflight > 2BDP), the flows will be restricted by CWND. When the shorter RTT flow and the longer RTT flow share the same bottleneck queue, the queuing delay will exceed the smaller *RTTmin* first. With the increase of *RTTmin*, the shorter RTT flow enters CWND-bounded mode, while the CWND-bounded flow cannot obtain additional bandwidth, even if more detections are carried out. After that, the queuing delay will not increase, and the longer RTT flow will never be restricted by CWND.

It can be seen from the analysis above that, when multiple BBR flows share the bottleneck link, the bottleneck link will be overloaded. Then the excess data packets are injected into the buffer, and a persistent queue is formed. The larger the BDP, the more packets can be sent to the buffer. The BDP of the long RTT flow is greater than that of the shorter flow, so the long RTT flow is dominant in the bottleneck queue and causes high latency [18]. According to the queuing theory, the increased queue share allows it to operate at a higher delivery rate than its competitor, so the longer RTT flow can occupy more bandwidth than the shorter RTT flow. With the increase of RTT difference of BBR flows, the fairness will deteriorate, and some users may take advantage of this vulnerability to deliberately increase latency in order to gain high bandwidth. Therefore, it is necessary to solve the problem of the longer RTT flow preference in BBR.

4.2. Model Analysis of BBR-ACW

In the ProbeBW phase, this loop detection will cause the total sending rate of multiple BBR flows to exceed the bottleneck capacity, then forming queuing delay. In addition, the buffer conditions on the bottleneck link should be considered. When the buffer is less than 1 BDP, the fixed *cwnd_gain* will make the packets sent by the BBR exceed the capacity of the link, and the packets will be lost due to buffer overflow.

Like BBR, BBQ and DA-BBR use a constant *cwnd_gain* to limit the transmission of each flow to 2 BDP. The CWND of the shorter flow is smaller than that of the longer flow. Thus, the shorter RTT flow will be constrained in advance and will gradually provide bandwidth for the longer RTT flow.

On the basis of simple experiments, the relationship between CWND and RTT fairness issue of BBR is analyzed. It is observed that the performance of different RTT flows increases with the increase of CWND. The shorter RTT flow is found to have higher bandwidth sharing, while the queuing delay is also increasing. The RTT fairness needs to be balanced against queuing delay.

Therefore, on the basis of original BBR, the *cwnd_gain* of different RTT flows is adaptively adjusted according to the delivery rate and RTT. Moreover, the impact of different buffer sizes is considered to avoid increased queuing delay or a large number of packet losses. This paper provides another way, which adaptively adjusts the upper limit of the CWND of each RTT flow and moves the operating point of BBR to the position where the buffer is not full.

Assume that the bandwidth of the bottleneck link L is C, and n flows are passing. Let $flow_i$ $i \in [1, n]$ denote flow i and d_i denote the delivery rate of $flow_i$. In an ideal state, $d_1 + d_2 + \ldots + d_n = C$. D_i represents the maximal delivery rate within 10 RTTs. Let $T_i = q_i + p_i$ denote the round-trip time of $flow_i$, where q_i is the queuing delay of $flow_i$ and p_i is *RTTmin* of $flow_i$. Let $I_i = d_i \times (q_i + p_i)$ denote the inflight of $flow_i$, which is the upper bound of the bottleneck link. Let $Q_i = d_i \times p_i$ represent the delivery capacity in the bottleneck link. The balance of the bottleneck link is determined by d_i and q_i. If $I_i > Q_i$, the inflight data in the bottleneck link will gradually fill up the buffer; if $I_i > 1.25\ Q_i$, the bottleneck link will be under high load, and packet loss may occur.

As shown in Algorithm 1, BBR-ACW updates I_i and Q_i through each probe and obtains the latest state of the bottleneck link, according to different RTT flows, to adjust the upper bound of CWND in the original BBR. When the link status becomes $I_i > Q_i$ with the link increasing, the inflight data will exceed the bottleneck delivery capacity. When it happens, the bottleneck link locks of additional capacity to transmit more data packets and forms a queue in the bottleneck buffer. Here, we let *min_rtt* indicate whether the lower *RTTmin* is measured within 3 RTTs. If no smaller *RTTmin* is obtained, *cwnd_gain* will adaptively adjust based on the D_i of different RTT flows, which will make the upper CWND be limited to reduce the inflight data in the probe up phase. When link status becomes $I_i > 1.25 \times Q_i$ and packet loss occurs (*has_loss* indicates), the network will be in an overload situation. Therefore, the *cwnd_gain* is reduced, according to the relationship between q_i and p_i, to adjust the number of packets sent in next cycle, and the CWND of the longer RTT flows should be reduced more to provide bandwidth for the shorter RTT flows. Otherwise, BBR-ACW will continue use the default *cwnd_gain* in the original BRR's *bbr_target_cwnd*.

Algorithm 1 Improvement of RTT Fairness:

INPUT: d_i, q_i, p_i, *has_loss*
$BDP = D_i \times p_i$
$CWND = cwnd\_gain \times BDP$
min_rtt = windowed_min(RTTmin,3RTTs)
  if $I_i <= 1.25\ Q_i$ **then**
    if *min_rtt > RTTmin*
    $cwnd\_gain = 2 \times \alpha$
    else
    $cwnd\_gain = 2$
  end if
else if $I_i > 1.25 \times Q_i$ **and** *has_loss* **then**
$cwnd\_gain = 2 \times \beta$
else
$cwnd\_gain = 2$
end if
return *cwnd_gain*

At the bottleneck link, all the BBR flows tend to send more data than expected, while some studies found that the longer RTT flow would inhibit the shorter RTT flow; thus, the *cwnd_gain* should be adjusted according to the RTT of each flow. Based on the original BBR algorithm, BBR-ACW takes the buffer constraints into account. According to d_i and T_i, the *cwnd_gain* is multiplied by the scale factor α or β to monitor the excess flow and to limit the inflight of different RTT flows within an adaptive multiple of CWND (the default is 2) to prevent BBR flows from injecting excessive data to the pipeline and increase the queuing delay.

If $I_i > Q_i$, the inflight data in the bottleneck link will gradually fill up the buffer. In order to improve the bandwidth utilization and prevent short RTT flows from being limited by CWND early, we add the scale factor α to increase the CWND. The scale factor α makes the nondominant flow to ensure at twice the original *cwnd_gain* (α is at least 1); The dominant flow can get 1–2 times the *cwnd_gain* (α is 1~2). Obviously, if the CWND is too large, it will lead to the increase of queuing delay, so that the value of α cannot be too large. The α value is set as Equation (1):

$$\alpha = \max\left[1, \frac{d_i - d_{min}}{d_{max} - d_{min}} \times 2\right] \tag{1}$$

where d_{max} and d_{min} are the maximum and minimum of delivery rate estimated observing under high load. We use the d_i to control α, and the d_i of the dominant flow is greater than that of the nondominant flow. It can be seen from the Equation (1) that the dominant flow

has a range of α from 1 to 2 and the nondominant flow has at least 1. It ensures that the nondominant flow will not be starved to death, and then the fairness between RTT flows is guaranteed. At the same time, it ensures the faster occupation of the available bandwidth and the convergence speed of BBR-ACW.

On the premise of ensuring RTT fairness, the speed of convergence fairness and link utilization are balanced. If $I_i > 1.25Q_i$, the bottleneck link will be under high load, and packet loss may occur. We hope to reduce the size of CWND and limit BBR flows. The scale factor β is introduced to appropriately reduce the CWND. The scale factor β can make the CWND of long RTT flow smaller than that of short RTT flow, and the bandwidth decreases faster. According to the queue backlog, the queue delay of long RTT flow is large, so we use T_i and p_i to control β. The value of β is set as Equation (2):

$$\beta = \frac{T_i}{T_i + p_i} \tag{2}$$

It can be seen from the Equation (2) that the *cwnd_gain* of the long RTT flow is smaller than that of the shorter RTT flow, which makes the difference between of the CWND sizes smaller than before. Due to the limitation of CWND, BBR-ACW can restrict different RTT flows to narrow the gap of data volume in bottleneck queue.

By the scale factor α and β, RTT unfairness can be alleviated, and bandwidth utilization can be improved. At the same time, the queuing delay caused by multiple data in the original BBR is reduced, and the queuing delay and retransmission are greatly reduced. This is obviously different from the previous approach to solving this problem by adjusting the BDP [30].

5. Experimental Evaluation and Analysis

In this section, BBR, BBQ, DA-BBR and BBR-ACW algorithms are implemented on NS3 platform, and simulation network topology is shown in Figure 3. We used BBR [2], BBQ [18] and DA-BBR [30] as benchmark algorithms and compared them with the BBR-ACW algorithm. The default active queue management (AQM) was DropTail policy, and every packet size was 1 kb. Different senders sent data to their corresponding receivers, and the bottleneck links between the routers they pass through were all the same.

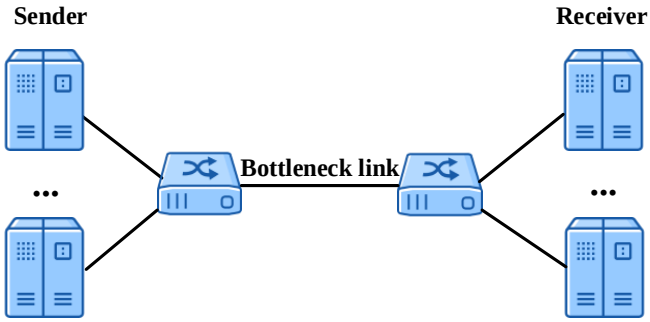

Figure 3. Experimental network topology.

The channel utilization, RTT fairness, retransmission and latency of the four algorithms were evaluated. To get more credible answers, the experiment was repeated 10 times for each test case and lasts 300 s.

5.1. Channel Utilization

In order to validate the performance of BBR-ACW, a channel utilization experiment was performed on a link with random packet loss. The buffer backlog configuration was

set as 1BDP or 5BDP; in both cases, the random packet loss rate was 0%, 1%, 3% and 6%, respectively. The channel utilization of all flows is calculated as Equation (3):

$$\eta = \frac{\sum_i bytes_i}{cap * duration} \tag{3}$$

where $bytes_i$ is the length of all received data packets on the application layer of flow$_i$. *Cap* is the bandwidth, and duration is the continuous simulation running time (300 s). The final result is shown in Figure 4.

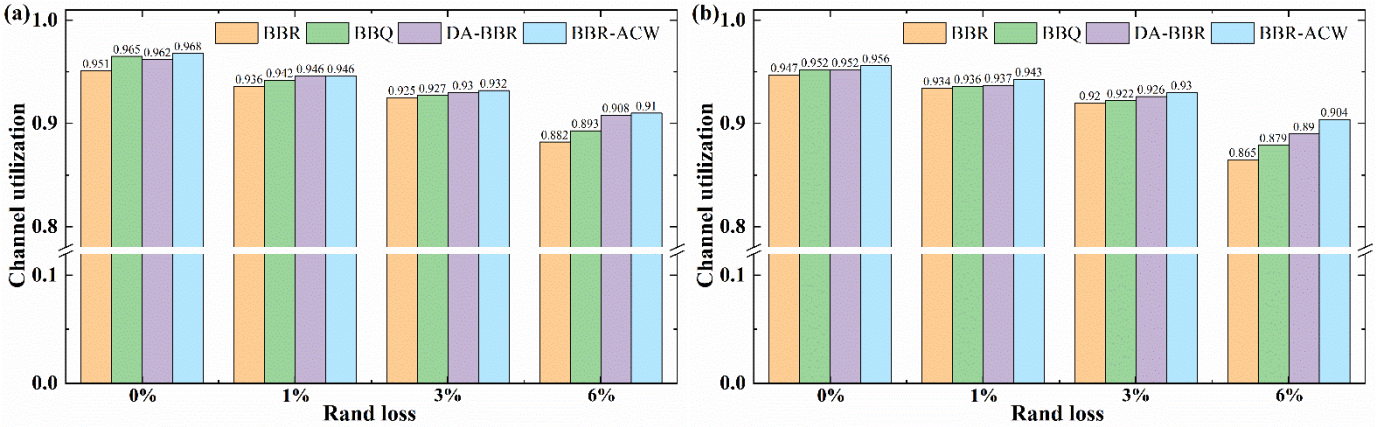

Figure 4. Channel utilization with random loss of different congestion control algorithms: (**a**) Bottleneck buffer size as 5 BDP (**b**) Bottleneck buffer size as 1 BDP.

When the random packet loss was 0%, these algorithms could get more than 94% channel bandwidth utilization with different buffer backlog. The BBR-ACW achieved the highest channel utilization, reaching about 97% in 5 BDP buffer and about 96% in 1 BDP buffer. The link utilization decreased as the random packet loss rate increased, but BBR-ACW was less affected by random packet loss. When the random packet loss rate was 6%, the channel utilization of BBR decreased significantly, while that of BBR-ACW, BBQ and DA-BBR decreased slightly. Moreover, BBR-ACW still maintained more than 90% channel utilization in different buffer sizes.

5.2. RTT Fairness

When the performance of the three algorithms, with respect to the RTT changes, was evaluated, flows with different RTTs preempted the 100 Mbps bottleneck bandwidth under the bottleneck buffer of 1 BDP or 5 BDP. The delay of flows were 10 ms and 50 ms flow, respectively, and both flows were running throughout the simulation time.

In the 5 BDP buffer, the throughput of different RTT flows of the four algorithms is shown in Figure 5. For BBR algorithm, although the 10-ms flow can quickly obtain bandwidth in the StartUp phase, as shown in Figure 5a, it is immediately suppressed by the 50 ms flow. After bandwidth stabilization, the average bandwidth of the 10 ms flow is only 8.9 Mbps, while the average bandwidth of the 50 ms flow is 85.2 Mbps. In Figure 5b, BBQ increases the average bandwidth occupied by the 10 ms flow, but the 50-ms flow still plays a dominant role, and the bandwidth of the 50 ms flow is twice larger than that of 10 ms flow. Figure 5c shows that, compared with the former two algorithms, DA-BBR improves the RTT fairness and reduces the throughput difference between 10 ms flow and 50 ms flow. In Figure 5d, BBR-ACW further improves the throughput of 10 ms flow. The bandwidth of 50 ms RTT flow is only 1.2 times that of 10 ms RTT flow, and the two flows can share the bandwidth equally.

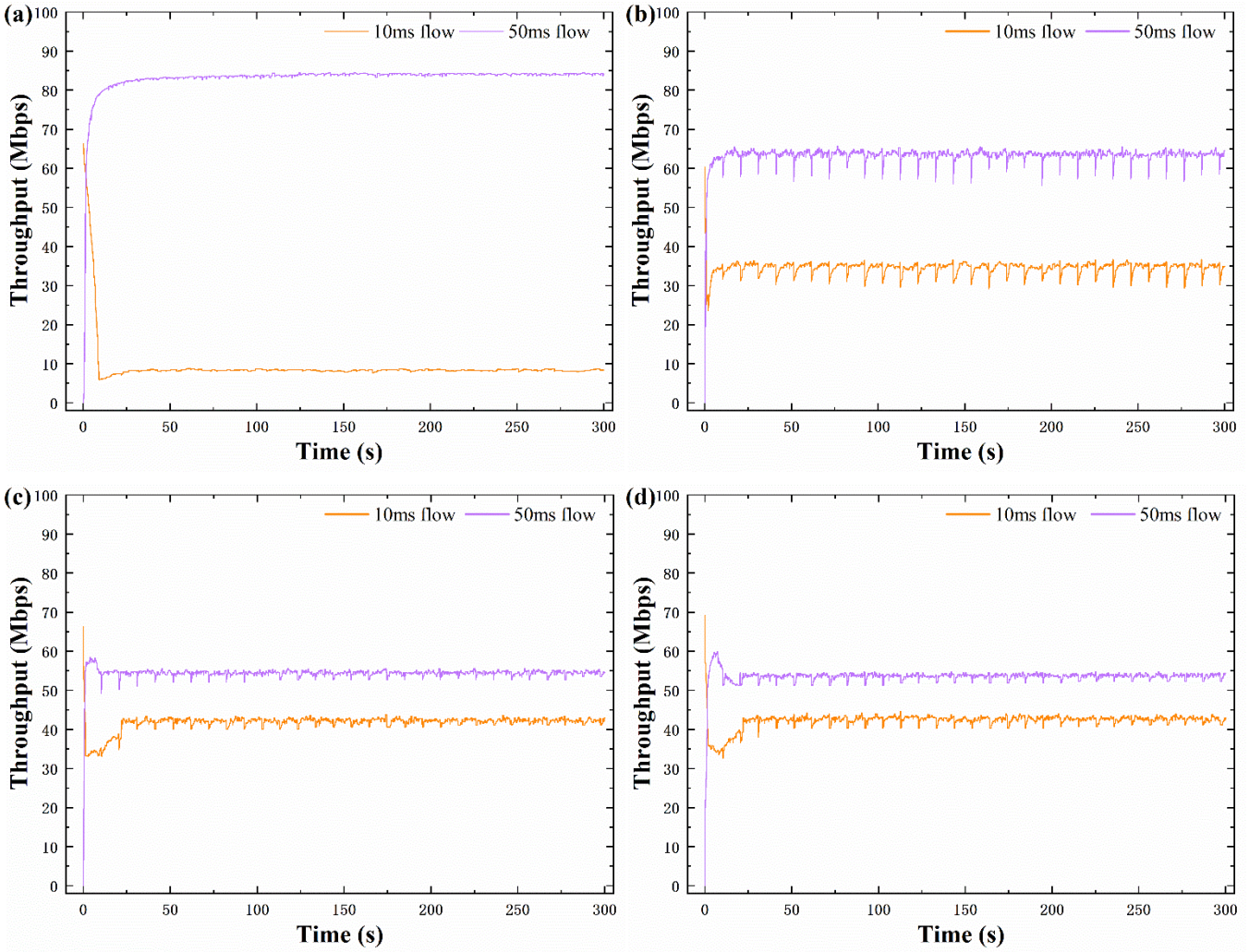

Figure 5. Throughput comparison with 5 BDP buffer: (**a**) BBR (**b**) BBQ (**c**) DA-BBR (**d**) BBR-ACW.

The above experiment was repeated again by us when the bottleneck buffer was reduced to 1 BDP, the result of which is shown in Figure 6. In Figure 6a, the throughput difference due to the BBR-RTT difference is still evident. The fairness of the BBQ algorithm is improved relative to that of BBR; as shown in Figure 6b, the improvement performance is reduced when compared with the large buffer (5BDP), and the difference between the two RTT flows is close to 3 times. In Figure 6c, compared with the other two algorithms above, the DA-BBR improve the bandwidth share of 10 ms RTT flow, but the 50 ms RTT flow still has the advantage. Figure 6d illustrates that the BBR-ACW provides more available bandwidth for 10 ms RTT flow than DA-BBR algorithm.

The extensive experiments were conducted under different conditions to quantify the improvement of RTT fairness by BBR-ACW, of which the impact of RTT disparity and buffer size on throughput bias were studied. The reference value of the average throughput is introduced for auxiliary analysis, and the calculation equation is shown in Equation (4).

$$\overline{x} = \frac{bytes}{duration} \tag{4}$$

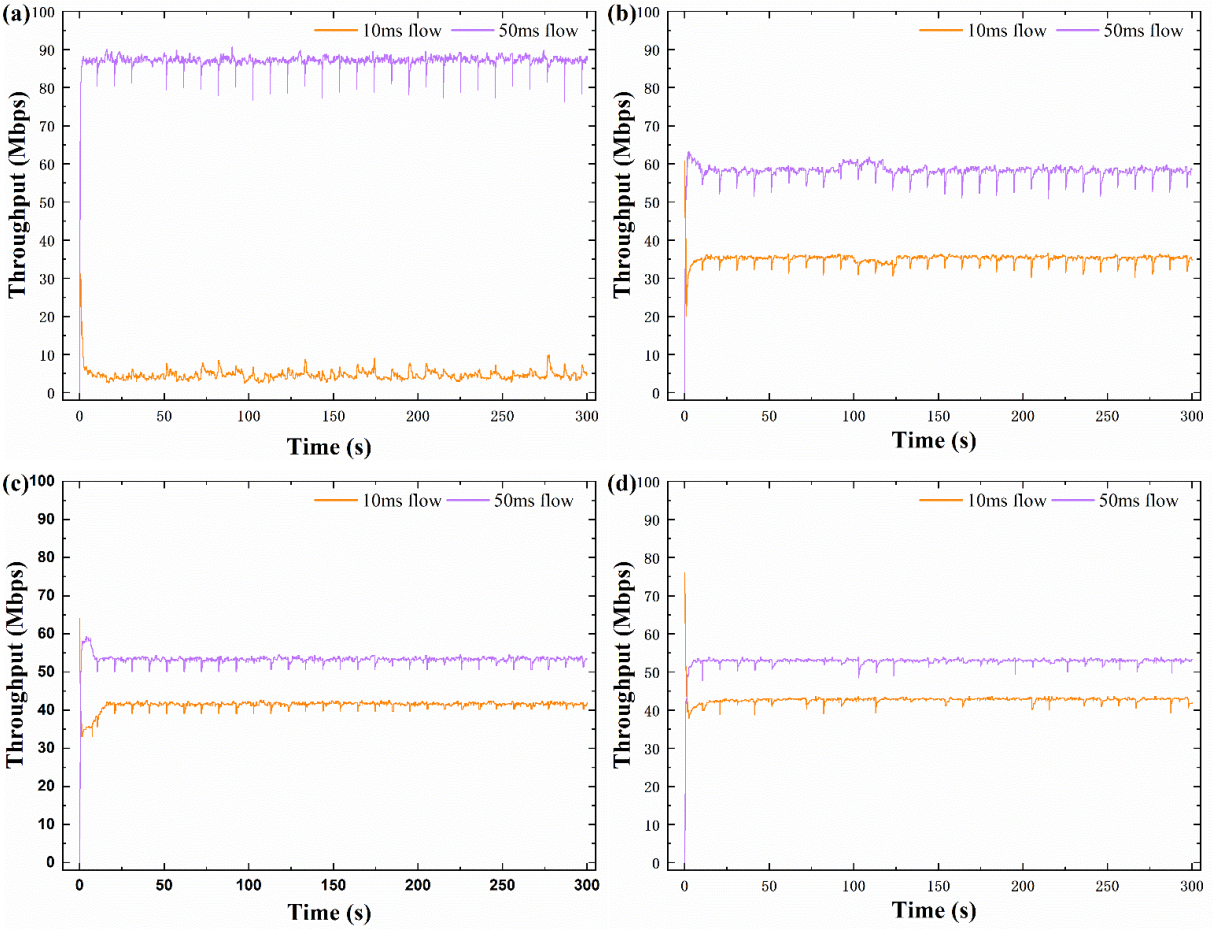

Figure 6. Throughput comparison with 1 BDP buffer: (**a**) BBR (**b**) BBQ (**c**) DA-BBR (**d**) BBR-ACW.

Figure 7a–d show the throughput changes of the four algorithms when 10ms RTT flow competes with other RTT flows in the 5BDP buffer, and Figure 7e–h illustrate which in the 1BDP buffer. As the RTT ratio increases in different cases, the fairness is decreased. For BBR flows in Figure 7a, the throughput difference is about 5.6 times, when the RTT difference is twice. When the RTT difference is greater than 3 times, the bandwidth of 10 ms RTT flow is only about 10%. The BBQ in Figure 7b shows the throughput of 10 ms flow does not depend on the constant change of RTT difference when the RTT difference is 2–3 times. When the RTT difference is 5 times, the throughput of long flow accounts about for 65%. In Figure 7c, the bandwidth ratio of long RTT flows are more than 58% from the point where the RTT difference is 5 times. For BBR-ACW algorithm, when RTT difference is less than 5 times, the long RTT flow can occupy less than 55% bandwidth occupation, and the 10-ms RTT flow can occupy more than 45%. Overall, BBR-ACW algorithm shows better fairness than the other three algorithms.

When the buffer size was 1BDP, RTT bias was reduced. As shown in Figure 7e, when the RTT difference is more than 2 times, the long RTT flows of BBR algorithm preempts about 80% of the bandwidth. In the BBQ algorithm shown in Figure 7f, the bandwidth of the long RTT flow still accounts about for 60% when the RTT difference is 5 times. As Figure 7g illustrates, DA-BBR shows better fairness than BBR and BBQ, similar to the case in the 5 BDP buffer. For BBR-ACW algorithm in Figure 7h, when different RTT flows compete, the fairness is slightly improved, compared with the DA-BRR algorithm. Even if it competes with 10 × RTT (100 ms), the 10-ms RTT flow can still retain close to 40% of the bottleneck bandwidth.

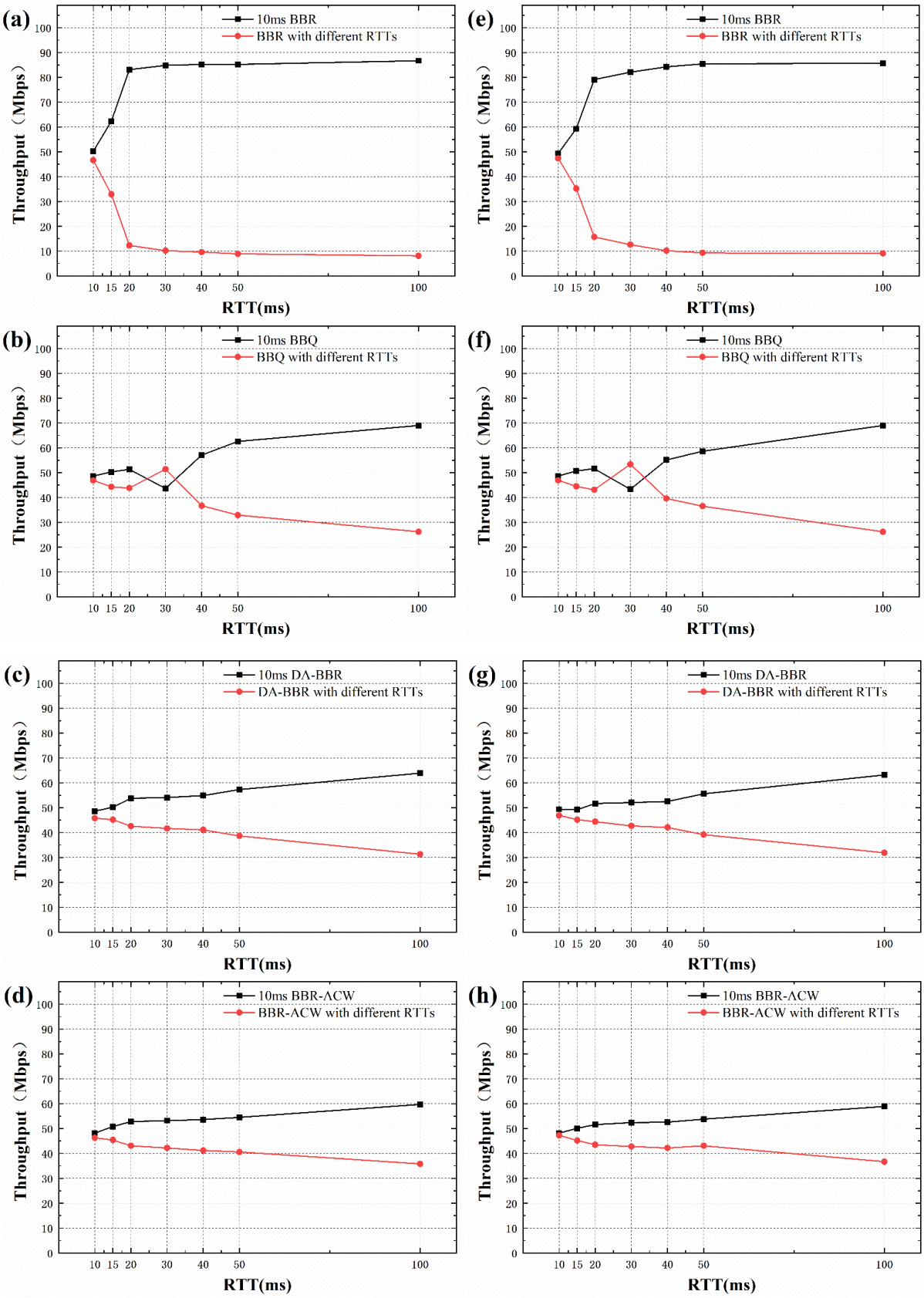

Figure 7. Throughput comparison when a 10-ms RTT flow coexists with a different RTT flow: (**a**) BBR with 1 BDP buffer. (**b**) BBQ with 1 BDP buffer. (**c**) DA-BBR with 1 BDP buffer. (**d**) BBR-ACW with 1 BDP buffer. (**e**) BBR with 5 BDP buffer. (**f**) BBQ with 5 BDP buffer. (**g**) DA-BBR with 5 BDP buffer. (**h**) BBR-ACW with 5 BDP buffer.

In some cases, the average throughput may not be a good reflection of the throughput difference. In order to further quantify the impact of the BBR-ACW algorithm on the fairness of RTT, the Jain's fairness index [36] was introduced, which indicates the fairness of bandwidth allocation when the competition for bandwidth resources happens. Equation (5) shows a method for calculating Jain's fairness index. The closer the value of J is to 1, the better the fairness of bandwidth allocation sharing.

$$J = \left(\frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} {x_i}^2} \right) \tag{5}$$

Figure 8 illustrates the Jain's index of the four algorithms in 10 Mbps, 50 Mbps and 100 Mbps bottleneck bandwidth, respectively. As shown in Figure 8a, when there is competition between 100 ms RTT flow and 10 ms RTT flow, the fairness index of BBR is the smallest, only about 0.60. Compared with BBR, the RTT fairness of BBQ is improved in different cases, and the fairness index is around 0.93 when RTT difference is less than 5 times. When the competition exists between 10 ms RTT and 100 ms RTT, the fairness index is 0.85. The fairness of DA-BBR is better than that of BBR and BBQ, and the minimum fairness index is about 0.92. The BBR-ACW's Jain index performs best, achieving a fairness index of 0.94, even when competing with a 100 ms RTT flow.

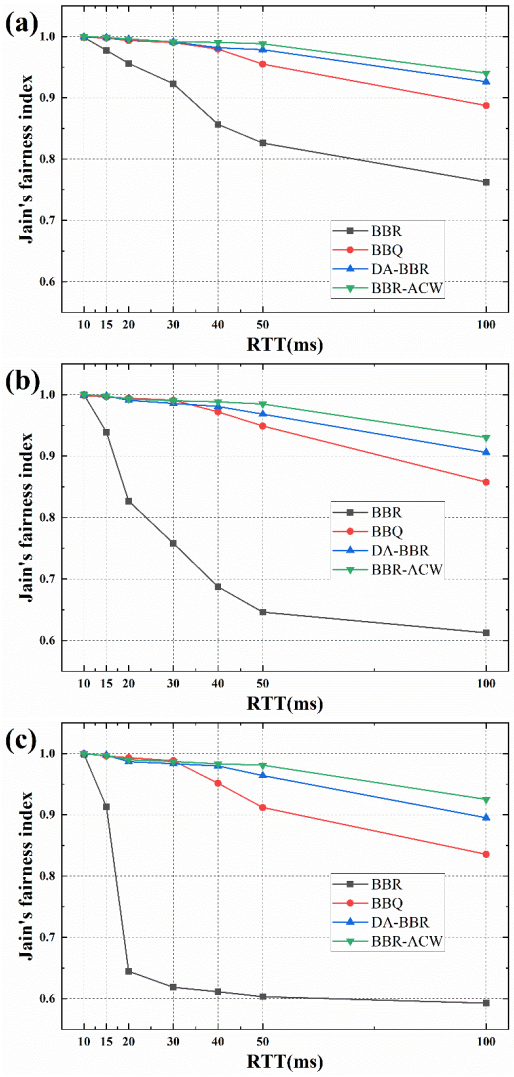

Figure 8. Jain's index comparison when a 10-ms round trip time (RTT) flow coexists with a different RTT flow: (**a**) 10 Mbps bottleneck bandwidth. (**b**) 50 Mbps bottleneck bandwidth. (**c**) 100 Mbps bottleneck bandwidth.

From Figure 8b, we can observe the fairness index decreases in different cases, when compared with Figure 8a. The minimum fairness index of BBR is only 0.61 as the RTT differences increase. When RTT difference is less than 3 times, the fairness indexes of BBQ, DA-BBR and BBR-ACW are above 0.94. When the RTT difference is more than 3 times, the fairness index of BBQ decreases significantly, and the minimum is only 0.85. The DA-BBR's fairness index is about 0.91. BBR-ACW demonstrates better fairness as compared to the other algorithms, and the fairness index is above 0.93.

For the 100 Mbps bottleneck bandwidth, as shown in Figure 8c, the fairness index of BBR is only 0.59. Although BBQ can improve the fairness index compared with BBR, which is not as good as in the condition of 10 Mbps bottleneck bandwidth, the fairness index only increases to 0.83 (the worst). Compared with the former two algorithms, the DA-BBR significantly improved the fairness of the different RTT flows, and the fairness index increased to about 0.9, but the fairness index of BBR-ACW was the highest among the four algorithms, and the minimum value was 0.92.

The BBR-ACW algorithm proposed in this paper obtained the largest fairness index among the four algorithms in different cases. Although the fairness index decreased when competing with 100-ms RTT stream, it still remained above 0.9, which represents a 5× fairness improvement, compared to the original BBR algorithm. In summary, BBR-ACW guaranteed better fairness in the different cases and increased the available bandwidth of 10-ms RTT flow.

5.3. Retransmission

Unlike the original BBR, BBR-ACW reduces the CWND when different RTT flows coexist. To check the effect of the buffer size on CWND adjustment, the different buffers were set to test the reduced number of retransmissions. The sender transmitted the 10-ms flows and 50-ms flows using different algorithms to the receiver with a bottleneck link bandwidth of 100 Mbps.

As shown in Figure 9, when the buffer size is 0.5 BDP, BBR generates a large amount of packet retransmission. BBQ and DA-BBR reduce packet retransmission, compared to the original BBR, but static *cwnd_gain* still causes buffer overflows. BBR-ACW has a smaller average CWND than BBR to avoid continuous packet loss, reducing retransmission by up to 78%. When buffer size is 1 BDP, the retransmission packet of BBR is 2992. BBQ reduces the number of retransmissions relative to BBR by 35%, and DA-BBR also reduces the number of retransmissions by 61%. BBR-ACW reduces retransmission by 63% and still has the lowest retransmission. BBR-ACW adaptively adjusted the *cwnd_gain* according to the size of bottleneck buffer and recognized packet loss as network congestion. Therefore, BBR-ACW reduced the overall transmission time and ensured the smaller number of retransmissions, regardless of buffer size.

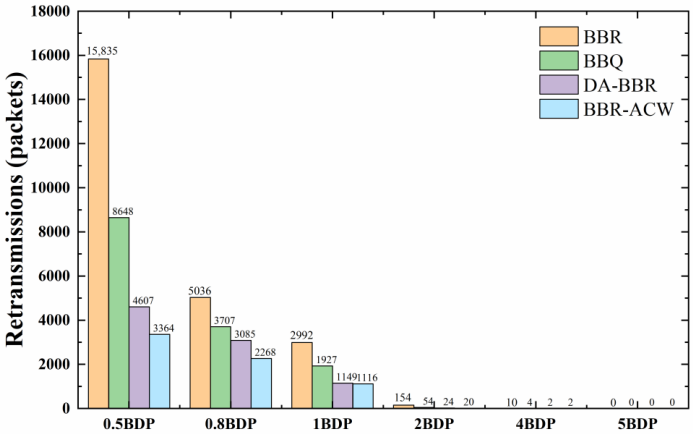

Figure 9. Retransmission with different RTT competition (10 ms vs. 50 ms) in different bottleneck buffers.

5.4. Latency

The RTT flow of 10 ms competed with the RTT flow of 50 ms with the bottleneck bandwidth of 100 Mbps and the bottleneck buffer size of 1 BDP. Figure 10 indicates the delay of 10 ms RTT flow with different algorithms. Figure 10a shows that the delay of BBR flow increases to 35 ms at most times. Figure 10b shows that the delay of BBQ flow is about 26 ms, which is 26% lower than the delay of BBR algorithm. As shown in Figure 10c, this is compared with the delay time of other two algorithms, in which DA-BBR is about 24 ms, which is 31% lower than the delay of BBR algorithm. Figure 10d shows that the delay of BBR-ACW flow is about 16 ms. BBR-ACW limited the aggregate flows inserted into the pipe, avoiding the high latency caused by it creating deep queue. Overall, the average queuing delay of BBR-ACW was 54% lower than that of BBR, which was also lower than that of BBQ and DA-BBR.

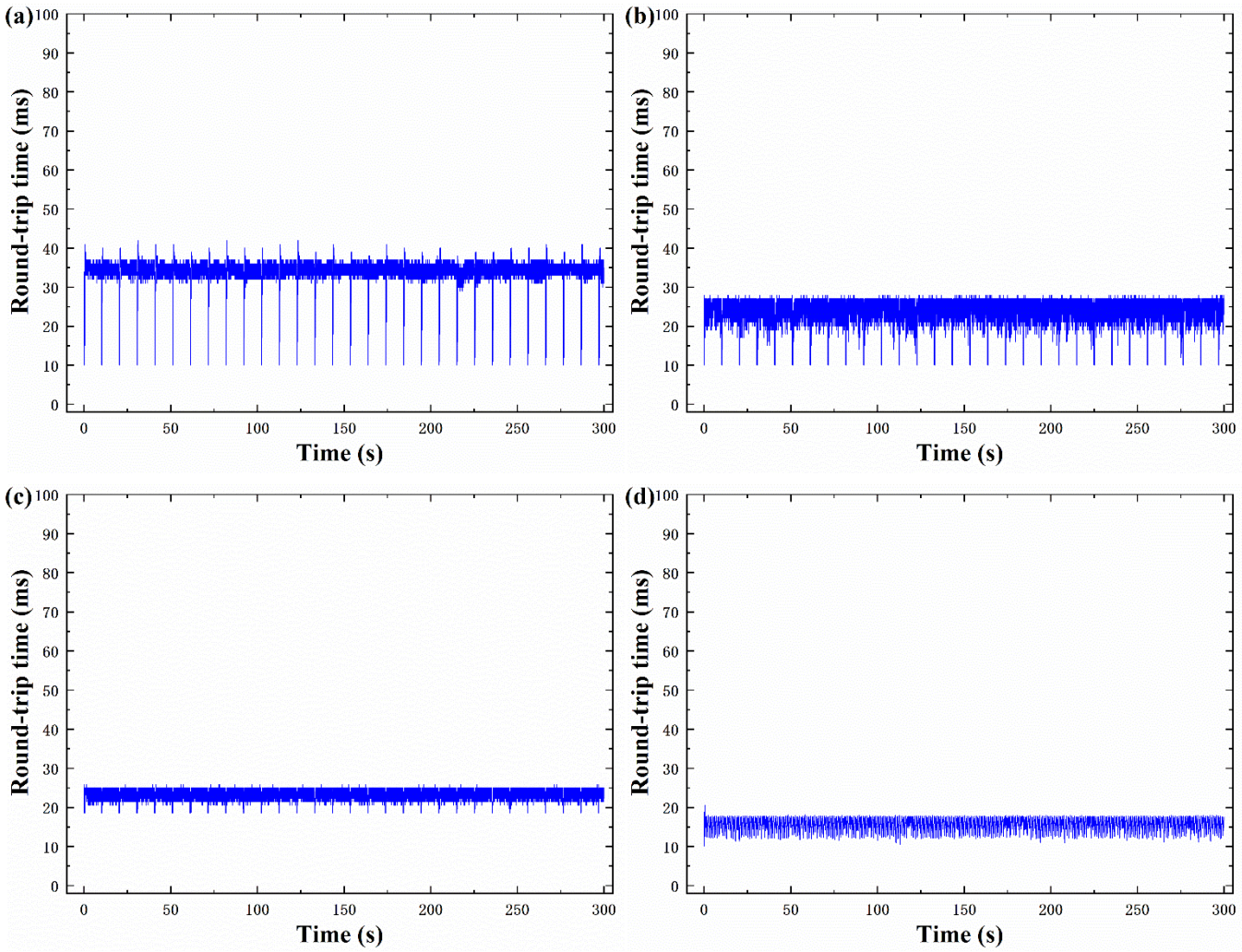

Figure 10. The latency of 10-ms RTT flow: (**a**) BBR; (**b**) BBQ, (**c**) DA-BBR and (**d**) BBR-ACW.

6. Conclusions

In this paper, based on the study of BBR congestion control, the cause of the unfairness of BBR to different RTT flows was analyzed. BBR-ACW was proposed to improve the fairness issue of original BBR with different RTT flows. According to the delivery rate and queue status of different RTT flows, the scaling factor α and β were defined to adjust the corresponding *cwnd_gain*. This scheme ensures that the different RTT flows can compete the bottleneck bandwidth fairly.

Many simulation experiments have been conducted on the NS3 platform for different RTT flows with different bottleneck bandwidths and buffer sizes. The results show that the performance of BBR-ACW is better than BBR, BBQ and DA-BBR in improving RTT fairness, and the fairness index is the highest of the four algorithms under different experimental conditions. Furthermore, BBR-ACW greatly reduced the number of retransmissions, compared to original BBR in the data transmission experiments. Lower packet retransmission can also be maintained even in small bottleneck buffers, which shows that the buffer size has a smaller restriction on BBR-ACW. In addition, the latency was also greatly improved, and the average queuing delay of BBR-ACW was lower than that of BBR, BBQ and DA-BBR.

In future work, what appropriate value to be chosen as the scaling factor requires data support in the real network, and it needs to be made applicable to BBR v2 if the same RTT fairness issue exists. We will continue the study of the BBR and BBR v2 algorithm and match the model with more actual data. Moreover, the AQM can be used to further improve BBR-ACW and provide better network performance.

Author Contributions: Methodology, W.P. and H.T.; data curation, W.P.; writing—original draft preparation, W.P.; writing—review and editing, X.L. (Xiru Li); supervision, X.L. (Xiaofeng Li); project administration, H.T. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China under Grant 61602435, in part by the National Natural Science Foundation of Anhui under Grant 1708085QF153.

Acknowledgments: The authors sincerely thank that the editors and reviewers for their valuable comments and suggestions on this article, which will comprehensively improve the quality of this article.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

BBR	Bottleneck Bandwidth and Round-trip time
BBR v2	BBR version 2
BBR-ACW	Adaptive Congestion Window of BBR
RTT	Round-trip time
BDP	Bandwidth Delay Product
BtlBw	Bottleneck Bandwidth
RTprop	Round-Trip propagation time
CWND	Congestion Window
DA-BBR	Delay-Aware BBR
NS3	Network Simulator 3
AQM	Active Queue Management

References

1. Jacobson, V. Congestion avoidance and control. *ACM Sigcomm Comput. Commun. Rev.* **1988**, *18*, 314–329. [CrossRef]
2. Floyd, S.; Henderson, T. *The NewReno Modification to TCP's Fast Recovery Algorithm*. 1999. Available online: https://tools.ietf.org/html/rfc2582 (accessed on 21 April 2020).
3. Xu, L.; Harfoush, K.; Rhee, I. Binary increase congestion control (BIC) for fast long-distance networks. In Proceedings of the IEEE Infocom, Hong Kong, China, 7–11 March 2004; Volume 4, pp. 2514–2524.
4. 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]
5. Cardwell, N.; Cheng, Y.; Gunn, C.S.; Yeganeh, S.H.; Jacobson, V. BBR: Congestion-based congestion control. *ACM Queue* **2016**, *14*, 50:20–50:53. [CrossRef]
6. Cardwell, N.; Cheng, Y.; Gunn, C.S.; Yeganeh, S.H.; Jacobson, V. BBR Congestion Control. In Proceedings of the IETF 97th Meeting, Seoul, Korea, 13–18 November 2016; Available online: https://www.ietf.org/proceedings/97/slides/slides-97-iccrg-bbr-congestion-control-02.pdf (accessed on 18 February 2020).
7. Kleinrock, L. Power and deterministic rules of thumb for probabilistic problems in computer communications. In Proceedings of the International Conference on Communications, Boston, MA, USA, 10–14 June 1979; pp. 1–10.

8. Cardwell, N.; Cheng, Y.; Gunn, C.S.; Yeganeh, S.H.; Jacobson, V. BBR Congestion Control: An Update. Available online: https://www.ietf.org/proceedings/98/slides/slides-98-iccrg-an-update-on-bbr-congestion-control-00.pdf (accessed on 2 April 2020).
9. Mahmud, I.; Kim, G.H.; Lubna, T. BBR-ACD: BBR with advanced congestion detection. *Electronics* **2020**, *9*, 136. [CrossRef]
10. Su, B.; Jiang, X.; Jin, G. Rethinking the rate estimation of BBR congestion control. *Electron. Lett.* **2020**, *56*, 239–241. [CrossRef]
11. Najmuddin, S.; Asim, M.; Munir, K.; Baker, T.; Guo, Z.; Ranjan, R. A BBR-based congestion control for delay-sensitive real-time applications. *Computing* **2020**, *102*, 2541–2563. [CrossRef]
12. Do, H.; Gregory, M.A.; Li, S. SDN-based Wireless Access Networks Utilising BBR TCP Congestion Control. In Proceedings of the 29th International Telecommunication Networks and Applications Conference (ITNAC), Auckland, New Zealand, 27–29 November 2019; pp. 1–8.
13. Wei, W.; Xue, K.; Han, J.; Xing, Y.; Wei, D.S.; Hong, P. BBR-based Congestion Control and Packet Scheduling for Bottleneck Fairness Considered Multipath TCP in Heterogeneous Wireless Networks. *IEEE Trans. Veh. Technol.* **2020**, *70*, 914–927. [CrossRef]
14. Jia, M.; Sun, W.; Wang, Z.; Yan, Y.; Qin, H.; Meng, K. MFBBR: An Optimized Fairness-aware TCP-BBR Algorithm in Wired-cum-wireless Network. In Proceedings of the IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Toronto, ON, Canada, 6–9 July 2020; pp. 171–176.
15. Cardwell, N. BBR v2: A model-based congestion control. In Proceedings of the ICCRG IETF 104th Meeting, March 2019. Available online: https://datatracker.ietf.org/meeting/104/materials/slides-104-iccrg-an-update-on-bbr-00 (accessed on 2 April 2020).
16. Cardwell, N.; Cheng, Y.; Yeganeh, S.H.; Jha, P.; Seung, Y.; Yang, K.; Swett, I.; Vasiliev, V.; Wu, B.; Hsiao, L.; et al. BBRv2: A Model-based Congestion Control. In Proceedings of the IETF 106th Meeting, Singapore, Singapore, 16–22 November 2019. Available online: https://datatracker.ietf.org/meeting/106/materials/slides-106-iccrgupdate-on-bbrv2 (accessed on 2 April 2020).
17. Hock, M.; Bless, R.; Zitterbart, M. Experimental evaluation of BBR congestion control. In Proceedings of the International Conference on Network Protocols (ICNP), Toronto, ON, Canada, 10–13 October 2017.
18. Ma, S.; Jiang, J.; Wang, W.; Li, B. Fairness of Congestion-Based Congestion Control: Experimental Evaluation and Analysis. *arXiv* **2017**, arXiv:1706.09115.
19. Scholz, D.; Jaeger, B.; Schwaighofer, L.; Raumer, L.; Geyer, F.; Carle, G. Toward a Deeper Understanding of TCP BBR Congestion Control. In Proceedings of the IFIP Networking, Zurich, Switzerland, 14–16 May 2018.
20. Atxutegi, E.; Haile, F.L.H.K.; Grinnemo, K.; Brunstrom, A.; Arvidsson, A. On the use of TCP BBR in cellular networks. *IEEE Commun. Mag.* **2018**, *56*, 172–179. [CrossRef]
21. Cao, Y.; Jain, A.; Sharma, K.; Balasubramanian, A.; Gandhi, A. When to use and when not to use BBR: An empirical analysis and evaluation study. In Proceedings of the Internet Measurement Conference, Amsterdam, The Netherlands, 21–23 October 2019; pp. 130–136.
22. Jaeger, B.; Scholz, D.; Raumer, D.; Geyer, F.; Carle, G. Reproducible measurement of TCP BBR congestion control. *Comput. Commun.* **2019**, *144*, 31–43. [CrossRef]
23. Ware, R.; Mukerjee, M.K.; Seshan, S.; Sherry, J. Modeling BBR's Interactions with Loss-Based Congestion Control. In Proceedings of the Internet Measurement Conference, Amsterdam, The Netherlands, 21–23 October 2019; pp. 137–143.
24. Kfoury, E.F.; Gomez, J.; Crichigno, J.; Bou-Harb, E. An emulation-based evaluation of TCP BBRv2 alpha for wired broadband. *Comput. Commun.* **2020**, *161*, 212–224. [CrossRef]
25. Miyazawa, K.; Sasaki, K.; Oda, N.; Yamaguchi, S. Cycle and Divergence of Performance on TCP BBR. In Proceedings of the IEEE International Conference on Cloud Networking (CloudNet), Tokyo, Japan, 22–24 October 2018.
26. Zhang, Y.; Cui, L.; Tso, F.P. Modest BBR: Enabling Better Fairness for BBR Congestion Control. In Proceedings of the IEEE Symposium on Computers and Communications (ISCC), Natal, Brazil, 25–28 June 2018.
27. Claypool, S.M. Sharing but not caring-Performance of TCP BBR and TCP CUBIC at the network bottleneck. In Proceedings of the Fifteenth Advanced International Conference on Telecommunications, Nice, France, 28 July—1 August 2019.
28. Song, Y.J.; Kim, G.H.; Cho, Y.Z. BBR-CWS: Improving the Inter-Protocol Fairness of BBR. *Electronics* **2020**, *9*, 862. [CrossRef]
29. Tao, Y.; Jiang, J.; Ma, S.; Wang, L.; Wang, W.; Li, B. Unraveling the RTT-fairness Problem for BBR: A queueing model. In Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 9–13 December 2018; pp. 1–6.
30. Kim, G.H.; Cho, Y.Z. Delay-Aware BBR Congestion Control Algorithm for RTT Fairness Improvement. *IEEE Access* **2019**, *8*, 4099–4109. [CrossRef]
31. Zhang, S. An Evaluation of BBR and its variants. *arXiv* **2019**, arXiv:1909.03673.
32. Yang, M.; Yang, P.; Wen, C.; Liu, Q.; Luo, J.; Yu, L. Adaptive-BBR: Fine-grained congestion control with improved fairness and low latency. In Proceedings of the 2019 IEEE Wireless Communications and Networking Conference (WCNC), Marrakech, Morocco, 15–18 April 2019; pp. 1–6.
33. Sun, W.; Jia, M.; Zhang, G.; Wang, Z. RFBBR: A Rtt Faireness Awared Algorithm Based on BBR. In Proceedings of the 2020 IEEE International Conference on Smart Internet of Things (SmartIoT), Beijing, China, 14–16 August 2020; pp. 124–131.
34. Kim, G.H.; Mahmud, I.; Cho, Y.Z. Fairness improvement of BBR congestion control algorithm for different RTT flows. In Proceedings of the 2019 International Conference on Electronics, Information, and Communication (ICEIC), Auckland, New Zealand, 22–25 January 2019; pp. 1–2.

35. Kim, G.H.; Song, Y.J.; Mahmud, I.; Cho, Y.Z. Enhanced BBR congestion control algorithm for improving RTT fairness. In Proceedings of the 2019 Eleventh International Conference on Ubiquitous and Future Networks (ICUFN), Zagreb, Croatia, 2–5 July 2019; pp. 358–360.
36. Jain, V.; Mittal, V.; Tahiliani, M.P. Design and implementation of TCP BBR in ns-3. In Proceedings of the 10th Workshop on Ns-3, Surathkal, India, 13–14 June 2018; pp. 16–22.
37. Jain, R.K.; Chiu, D.M.W.; Hawe, W.R. *A Quantitative Measure of Fairness and Discrimination*; Eastern Research Laboratory, Digital Equipment Corporation: Hudson, MA, USA, 1984.
38. Jaffe, J. Flow control power is nondecentralizable. *IEEE Trans. Commun.* **1981**, *29*, 1301–1306. [CrossRef]