Adaptive-BBR: Fine-Grained Congestion Control with Improved Fairness and Low Latency

Ming Yang*, Peng Yang*, Chaozhun Wen*, Qiong Liu*, Jingjing Luo†, and Li Yu*
*School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, China
†School of Electronic and Information Engineering, Harbin Institute of Technology (Shenzhen)
Email: {m_yang, yangpeng, czwen}@hust.edu.cn, {liuqiong2018, luojingjing1989}@gmail.com, hustlyu@hust.edu.cn

***Abstract*—Traditional loss-based congestion control protocols interpret packet loss as network congestion. Recently, Google proposed BBR, which is a congestion-based congestion control protocol. It employs delivery rate as the knob for congestion control, which achieves higher throughput and lower latency. Interestingly, BBR is found to have a preference for longer round-trip time (RTT) flows, which enjoy higher bandwidth ratio compared to flows with shorter RTT. To address this fairness issue, we proposed Adaptive-BBR, which creatively uses adaptive pacing gain to adjust the sending rate. The objective is that, via the proposed fine-grained adaptive mechanism, flows with different RTTs share similar portion of bottleneck bandwidth. Simulation results show that Adaptive-BBR can improve fairness by at least 47.8%, and reduce average queuing delay by up to 93.3%, compared with that of BBR.**

I. Introduction

In order to fully utilize network resources while ensuring minimal link congestion, typical loss-based congestion control (e.g., TCP NewReno [4]) uses additive increase multiplicative decrease mechanism to adjust each flow's congestion window (CWND). The sender will increase its sending window size by one unit during every RTT if there is no packet loss. Once packet loss is detected (via duplicated ACKs), the CWND will be set to half of the current window size. However, with the evolving network devices and infrastructure, interpreting packet loss as network congestion is not always true, especially in the case of small bottleneck buffer sizes [5]. As a result, loss-based congestion control results in low throughput and high latency. For delay-sensitive applications, such as online gaming and virtual reality video streaming, high latency will significantly deteriorate user experience [6], [7].

Alternative to loss-based congestion control, Google recently proposed congestion-based control (BBR) [5], which aims to work at the optimal operating point (maximum delivery rate while minimum delay) by adjusting its sending behavior based on the estimated bottleneck bandwidth (BtlBw) and round-trip propagation time (RTprop). The implementation in various Google services shows that BBR has notable advantages compared with CUBIC, in terms of throughput and delay. Yet, BBR has an obvious preference for long RTT flow. Longer RTT will lead to higher bandwidth-delay product, which triggers higher sending rate for larger volume of in-flight packets. Nevertheless, it is of significant importance to fairly allocate network resource for multiple connections. To improve the fairness of congestion control, Ma *et al.* proposed BBQ to limit the number of packets sent by long RTT flow during probing period [9]. However, the root cause of BBR unfairness is as follows. During steady state, BBR explores available bandwidth without considering the share of different flows on the bottleneck. Then, each flow blindly and asynchronously explores bandwidth, resulting in the mismatch between aggregated sending rate and the BtlBw. Such mismatch issue remains unresolved in BBQ, hence the effect of BBQ is limited and flows still experience high latency.

In this paper, in order to tune flow sending rate with the knowledge of bottleneck buffer information in the BBR framework, we propose Adaptive-BBR. The core idea is to introduce adaptive pacing gain to adjust sending rate based on the current flow sharing information of BtlBw. Before exploring bandwidth, Adaptive-BBR will determine whether current bottleneck capacity can support additional flows' *inflight* (data sent but not yet acknowledged) or not, through comparing current *inflight* with delivery capacity (delivery rate-RTprop product). Then Adaptive-BBR adopts aggressive pacing gain to explore bandwidth only when bottleneck capacity can deliver current *inflight* packets. Queuing analysis indicates that backlog determines bandwidth share of competing flows. Accordingly, an *Pacing Gain Update* mechanism (Algorithm 1) is designed to choose adaptive pacing gain that makes different flows keep equal and small amount of data in the bottleneck queue. To evaluate the performance of the proposed algorithm, we implement Adaptive-BBR using ns-3 and compare with BBR and BBQ in terms of fairness, latency as well as responsiveness. The results indicate Adaptive-BBR outperforms others with improved fairness and lower latency without obvious degradation of responsiveness. In particular, when two flows compete for 20 Mbps bandwidth, Adaptive-BBR can reduce the queuing latency by up to 93.3%, compared with BBR. By configuring different bottleneck bandwidth, RTT disparity and the number of competing flows during extensive experiments, it is demonstrated that Adaptive-BBR improves the Jain index [11] of BBR by at least 47.8%. The main contribution of this paper can be summarized as follows.

- We find that the BBR unfairness is due to the fact that flows are blindly and asynchronously exploring bandwidth, which results in the mismatch between aggregated sending rate and BtlBw.
- We propose Adaptive-BBR to address the BBR fairness

issue with fluid model analysis. By choosing adaptive pacing gain that adjusts sending rate based on the current sharing of bottleneck bandwidth of different flows, Adaptive-BBR improves fairness and reduce latency.

- We conduct extensive experiments to validate the performance of the proposed Adaptive-BBR via ns-3. The results show that Adaptive-BBR outperforms BBR and BBQ with improved fairness and lower latency.

The remainder of the paper is organized as follows. Section II reviews the related work. Section III briefly introduces BBR and analyzes the rate mismatch issue during steady phase. In Section IV, we propose Adaptive-BBR with fluid model analysis. Section V presents extensive ns-3 experimental results of the proposed Adaptive-BBR by comparing to BBR and BBQ. Finally, Section VI concludes this paper.

II. Related Work

There have been many schemes proposed to avoid network congestion [2], [3]. Traditional loss-based congestion control (e.g., CUBIC [1]) considers packet loss as the signal of congestion. When buffer size is large, CUBIC pushes the volume of in-flight packet to the buffer limit, which results in unnecessary high latency called *bufferbloat* [12]. Interestingly, as the default congestion control in Windows, Compound TCP considers packet loss and delay as the signal of congestion, but its performance will be affected by active queue management (AQM) variants and buffer size [17]. Although AQM (such as CoDel [18]) can reduce queuing latency to improve the performance of congestion control by dropping packet before buffer is exhausted, currently there is no AQM schemes that can suit general congestion control policies.

BBR was originally introduced by Google in [5]. Hock *et al.* conducted the first evaluation and tested the performance of BBR at a high-speed bottleneck link [8], including RTT-fairness, latency, inter-protocol with CUBIC and packet loss. It was shown that BBR favors long RTT flow when multiple flows sharing a bottleneck link with large buffer size. With smaller bottleneck buffer, BBR flows will severely undergo packet loss and high retransmission ratio. In addition, CUBIC flow will suppresses the BBR flow if the buffer size is large. Conversely, BBR flow will starve CUBIC flow when the buffer size is small. Ma *et al.* [9] found the RTT-fairness problem of BBR, and propose BBQ to improve the fairness of BBR. They observed that the short RTT flow is constrained by congestion window and long RTT flow will dominate the bottleneck buffer. To improve the bandwidth occupied by short RTT flow, BBQ limits the number of packets injected into the pipe by long RTT flow using $min(\alpha, RTT_{min})$ as the duration of probing period, when $RTT > (1+\beta)RTT_{min}$.

The root cause of the problem is that the aggregate sending rate fails to match the bottleneck bandwidth, short RTT flow is still more likely to be limited by CWND. Hence, BBQ flows still undergo high latency. Different from BBR and BBQ, which use fixed pacing gain, Adaptive-BBR will choose adaptive pacing gain according to the current condition of bottleneck link. And all flows are restricted to similar but small amount of data in bottleneck queue by adjusting their sending rate based on adaptive pacing gain.

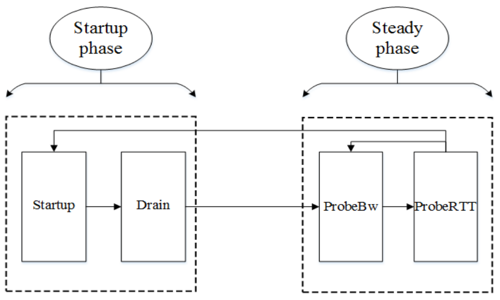


Fig. 1. The operating mechanism of BBR.

III. Introduction and Analysis of BBR's Steady Phase

According to the optimum operating point [5], the product of BtlBw and RTprop is named as *BDP*, which determines the amount of *inflight* that can be sent into the pipe to obtain the maximal delivery rate and minimum latency simultaneously. The purpose of BBR is to adjust the amount of *inflight* to current *BDP* by controlling the sending rate. As illustrated in Fig. 1, BBR algorithm's state machine consists of Startup phase and Steady phase [5]. Every phase contains two process. It uses the maximum measurement of delivery rate $\widehat{S}$ in the past 10 RTTs as the current BtlBw. Similarly, it obtains the current RTprop by detecting the minimum measurement of RTT ($\widehat{RTT}_{min}$) in the past ten seconds. Startup process is similar to the slow start process of other congestion control algorithms, followed by a Drain process to evacuate the queue accumulated during the former.

A. BBR Steady phase

After Startup phase, BBR gets $\widehat{S}$ and $\widehat{RTT}_{min}$, then it enters ProbeBw state. During ProbeBw, the dynamic $\widehat{S}$ of bottleneck link will be updated through a cyclic probing mechanism. Every cycle contains eight periods, and BBR adjusts sending rate on the basis of current $\widehat{S}$ in every period by using different pacing gain. The duration of every pacing gain is $\widehat{RTT}_{min}$.

In the first period, BBR chooses to send 1.25*BDP* packets [5] at the speed of $1.25\widehat{S}$ (pacing gain = 1.25) to explore more available bandwidth. New delivery rate can be calculated and stored into the current window by BBR through ACK signal's feedback. $\widehat{S}$ will be updated to the new delivery rate if it is higher than the current $\widehat{S}$. In the second period, BBR will send 0.75*BDP* packets at the sending rate $0.75\widehat{S}$ (pacing gain = 0.75), expecting to drain excess packets caused by unchanged $\widehat{S}$ in the former period. Once the amount of *inflight* decreases to *BDP*, this period terminates. Next, BBR will steadily send the *BDP* packets at the speed of $\widehat{S}$ (pacing gain = 1).

BBR runs iteratively. Once all RTTs in the past ten seconds are larger than $\widehat{RTT}_{min}$, the current $\widehat{RTT}_{min}$ expires and BBR enters ProbeRTT to re-estimate the actual RTprop.

During ProbeRTT, BBR only sends four packets during each RTT to ensure there is no queuing delay interferes. When PropRTT is completed, BBR enters either Startup or ProbeBw, depending on whether the pipe is full or not. Although BBR does not control the sending rate via congestion window, it sets the upper bound of BBR flow's *inflight* to 2*BDP* [5].

B. Analysis of Steady phase

Assume that there are n BBR flows traversing the bottleneck link with bandwidth C. Let $flow_i$ ($i \in [1, n]$) denote the flow *i* and $d_i(t)$ represent the delivery rate of $flow_i$ at time *t*. $\widehat{S}_i(t) = max(d_i(T))$ ($\forall T \in [t - 10RTT, t]$) denotes the maximal delivery rate of $flow_i$ at time *t*.

In the case of $n = 1$, only one flow passes through the bottleneck link. Since delivery rate is upper-bounded by BtlBw after startup phase, the flow's $\widehat{S}_i(t)$ is equal to BtlBw. BBR flow will converge to the optimal operating point, operating with maximum delivery rate and minimum delay.

In the case of $n > 1$, after startup, BtlBw has been detected. When they enter ProbeBw, the bottleneck link is fully utilized, and $\sum_{i=1}^{n} d_i(T) = C$. According to $\widehat{S}_i(t) = max(d_i(T))$, it can be deduced that $\sum_{i=1}^{n} \widehat{S}_i(t) \geq \sum_{i=1}^{n} d_i(T) = C$. If and only if $\widehat{S}_i(t) = d_i(T_0)$ ($\forall i \in [1, n]$), the equal sign holds. However, due to different RTT and flow's random accession, BBR flows will asynchronously increase sending rate to explore available bandwidth, which lead to inaccurate $\widehat{S}_i(t)$ estimation. In other words, average sending rate of all flows will be higher than BtlBw. Even after certain flows terminate, the above conclusion still holds, as the remaining bandwidth will be quickly occupied by other flows.

The above analysis indicates that the bottleneck link will be overloaded when it is shared by multiple BBR flows. Since aggregated sending rate fails to match BtlBw, excess packets will be injected to the buffer, leading to high latency. Long RTT flow has a larger *BDP* than the short ones, so it can insert more packets into the pipe and occupies a larger proportion of the bottleneck queue [9]. According to queuing theory, long RTT flow will occupy more bandwidth than the short ones.

IV. The Proposed Adaptive-BBR

For BBR, available bandwidth occupied by one flow is solely determined by physical bottleneck link and the influence of other flows is ignored. However, when there are multiple flows, original probing mechanism will cause aggregate sending rate exceed bottleneck's capacity, which is the root cause of BBR unfairness and high latency. We propose Adaptive-BBR that adopts adaptive pacing gain to improve fairness and reduce latency.

A. Adaptive-BBR

Unlike BBR and BBQ, which blindly probe the bandwidth through aggressively sending excessive packets without considering the buffer condition on bottleneck link, Adaptive-BBR will apply adaptive pacing gain to flexibly adjust the sending rate based on bottleneck buffer information. Meanwhile, Adaptive-BBR will carefully and properly use 1.25 pacing gain to avoid sending excessive packets.

Algorithm 1 Pacing Gain Update Algorithm

```
Input: $\widehat{S}_i$, $\widehat{RTT}_{min}$, $I_i$, $d_i$ of $flow_i$
Output: pacing gain $PG_i$
 1: $\alpha = 3$ * packet size;
 2: $D_i = \widehat{RTT}_{min} * d_i$;
 3: $Q_i = I_i - D_i$;
 4: $BDP = \widehat{S}_i * \widehat{RTT}_{min}$;
 5: for every ACK do
 6:     if $Q_i == 0$ then
 7:         $PG_i$ = 1.25;
 8:     else if $Q_i > \alpha$ then
 9:         $PG_i$ = $1 - \{\frac{Q_i - \alpha}{8BDP}\}$;
10:     else
11:         $PG_i$ = $1 + \{\frac{Q_i - \alpha}{8BDP}\}$;
12:     end if
13: end for
14: return $PG_i$
```

Let $D_i(t) = \widehat{RTT}_{min} * d_i(t)$ denote the delivery capacity of bottleneck link of $flow_i$ at time t. Let $I_i(t)$ be the amount of $flow_i$'s *inflight* at time *t*. Let $Q_i(t) = I_i(t) - D_i(t)$ denote the disparity between $I_i(t)$ and $D_i(t)$. Since I_i is the upper bound of data (D_i) delivered by bottleneck link, $Q_i(t) \geq 0$.

If $Q_i(t) = 0$, then the bottleneck link can support $flow_i$'s *inflight*. In the next period, $flow_i$ is able to send more packets to occupy the available bandwidth. Only in this case, Adaptive-BBR adopts 1.25 pacing gain to quickly occupy available bandwidth. If $Q_i(t) > 0$, $flow_i$'s *inflight* exceeds the delivery capacity of bottleneck link. To reduce bottleneck's load, $flow_i$ should decrease the number of packets inserted into the pipe in next period.

As shown in Algorithm 1, Q_i gets update every ACK and reflects the dynamic state of bottleneck link. When $Q_i(t) = 0$, it indicates bottleneck link currently has additional capacity to delivery more packets for $flow_i$. As a result, Adaptive-BBR will use an aggressive pacing gain of 1.25 to occupy available bandwidth. As sending rate increases, growing *inflight* packets will fill the bottleneck buffer. In this case, $Q_i(t) > \alpha$ holds. Then the value of pacing gain is calculated through corresponding Q_i (pacing gain equals 1 + $\frac{\alpha - Q_i}{8BDP}$). According to queuing theory, bandwidth allocation will be determined by the backlog of persistent queue. Then, the existence of α, which is an identical but small amount (3 * packet size), can restrict all flows to keep similar amount of data in bottleneck queue to improve fairness. In addition, considering responsiveness and route change, Adaptive-BBR experimentally sets the expired time of $\widehat{RTT}_{min}$ as 150 s.

B. Fluid Model Analysis of Adaptive-BBR

Consider a network with a set of links *L*, and there is a bottleneck link l_b with bandwidth C. The bottleneck link is shared by *N* flows, and $flow_i$ passes through a set of links

L_i ($L_i \subseteq L$). Let $T_i(t) = q_i(t) + p_i$ denote the round-trip time of $flow_i$ at time *t*. Queuing delay and RTprop of $flow_i$ are respectively denoted by $q_i(t)$ and p_i. At time *t*, each flow adjusts its sending rate $x_i(t)$ according to $Q_i(t)$ and the maximum delivery rate $\widehat{S}_i(t)$.

$$I_i(t) = d_i(t) * T_i(t) = Q_i(t - p_i) + x_i(t) * p_i \quad (1)$$

$$Q_i(t) = I_i(t) - D_i(t) = d_i(t) * q_i(t) \quad (2)$$

The dynamics of *inflight* is

$$\dot{I}_i(t) = x_i(t) - d_i(t - p_i), \quad (3)$$

and the ACK rate's dynamic of $flow_i$ is

$$\dot{d}_i(t) = (\frac{x_i(t) - d_i(t - p_i)}{T_i(t)})^{+}_{d_i(t)}, \quad (4)$$

where $(a)^{+}_{b} = a$, if $a > 0$ or $b > 0$. Otherwise, $(a)^{+}_{b} = 0$. According to the *Update Pacing Gain*, $x_i(t)$ is defined as:

$$x_i(t) = \widehat{S}_i(t) * (1 + (\frac{\alpha - Q_i(t)}{8BDP})) \quad (5)$$

$$BDP = \widehat{S}_i(t) * p_i \quad (6)$$

Substituting (5) and (6) into (4), we have

$$\dot{d}_i(t) = (\frac{8BDP + \alpha - d_i(t)q_i(t) - 8d_i(t - p_i)p_i}{8p_iT_i(t)})^{+}_{d_i(t)} \quad (7)$$

Equation (7) can be rewritten as

$$\dot{d}_i(t) = K_i(d_i)(V_i(d_i) - q_i(t))^{+}_{d_i(t)} \quad (8)$$

From (7), $K_i(d_i)$ and $V_i(d_i)$ can be obtained

$$K_i(d_i) = (\frac{d_i(t)}{8p_iT_i}) \quad (9)$$

$$V_i(d_i) = (\frac{8BDP + \alpha - 8d_i(t - p_i)p_i}{d_i(t)}). \quad (10)$$

$K_i(d_i)$ influences the responsiveness and window oscillation, and $V_i(d_i)$ decides the equilibrium of congestion control algorithm [10]. When $\dot{d}_i = 0$, the equilibrium d_i^* is

$$d_i^* = (\frac{8BDP + \alpha}{8p_i + q_i^*}). \quad (11)$$

Adaptive-BBR adjusts flow's sending behavior during steady phase based on the current *BDP*. Equation (11) indicates that the equilibrium d_i^* will be determined by $\widehat{S}_i$ and $q_i{}^*$. $\widehat{S}_i(t)$ is defined as the maximum delivery rate during a window of ten RTTs, and it will be influenced by the RTT disparity of flows during startup and drain process. Generally, short RTT flow will occupy and maintain a larger $\widehat{S}_i(t)$ than other flows during startup process. However, when short RTT flow enters drain process, long RTT flows may still stay in startup process, which means they can send more packets to obtain a larger $\widehat{S}_i(t)$. Which process dominates depends on the disparity of RTT. In addition, $q_i{}^*$ is proportional to the flows' number. Although Adaptive-BBR is getting weaker in reducing latency as the number of flows increases, the delay caused by BBR is the upper bound of that caused by Adaptive-BBR due to CWND limitation.

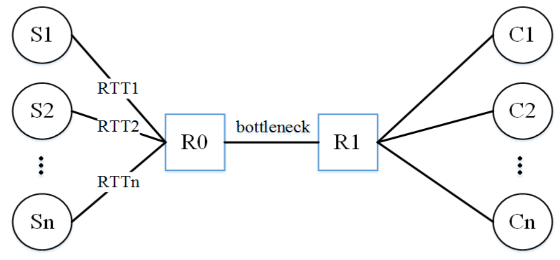


Fig. 2. Dumbbell topology.

V. Performance Evaluation

In this section, we evaluate the performance of our Adaptive-BBR using ns-3, and compare it with BBR and BBQ in terms of RTT-fairness, latency and responsiveness.

A. Simulation Setup

In the following simulations, every packet size is 1 kb and the default AQM is DropTail. Fig. 2 shows the simulation topology. Sender S_n sends the data to the corresponding receiver C_n, which passes through the same bottleneck link between router R0 and router R1. For evaluating BBQ, we adopt $\alpha = 3ms$, $\beta = 0.8\%$ [9].

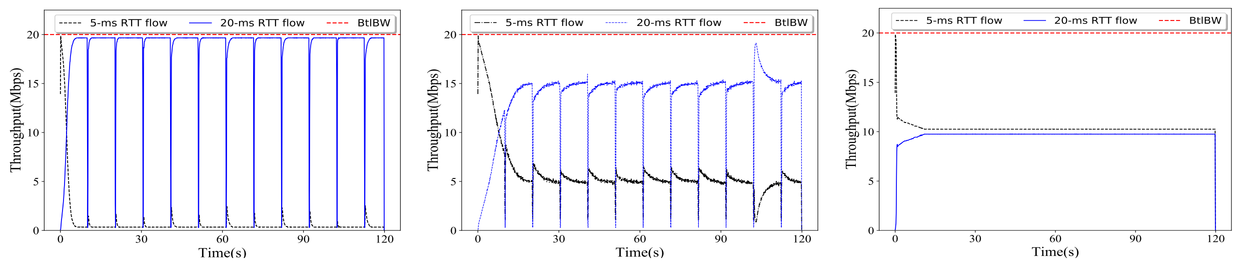
B. RTT-fairness

To evaluate the performance of RTT-fairness, two flows with different RTT will compete for 20 Mbps bottleneck bandwidth. Bottleneck buffer is 5 MB, which is much larger than the *BDP*. As shown in Fig. 3(a), although 5-ms BBR flow can quickly get the bandwidth before 3 s, 20-ms BBR flow will always suppress and starve it in the later period. Since aggregate sending rate mismatches BtlBw, 5-ms BBR flow is earlier constrained by the smaller CWND, and cannot insert more packets into the pipe. Finally, 5-ms BBR flow only occupies average 0.8 Mbps, but 20-ms BBR flow occupies average 18.8 Mbps. Fig. 3(b) indicates BBQ can improve the average bandwidth occupied by 5-ms RTT flow (around 5.8 Mbps) by limiting the amount of data send by 20-ms RTT flow during probing period. However, the effect of BBQ is limited and 20-ms RTT flow still dominates (around 13.8 Mbps), due to high latency. As Fig. 3(c) shows, Adaptive-BBR can obtain better performance of fairness. 5-ms RTT flow (around 10.2 Mbps) and 20-ms RTT flow (around 9.7 Mbps) will steadily occupy similar bandwidth from 13 s.

In order to quantify the RTT-fairness, Jain's Fairness Index is introduced, which is widely used as a metric of network-wise fairness performance in network throughput allocation [13]–[16] and defined as follows

$$J = (\frac{(\sum_{i=1}^{n} d_i)^2}{n\sum_{i=1}^{n} d_i^2}) \quad (12)$$

The range of J is $[\frac{1}{n}, 1]$ and d_i is the average throughput of $flow_i$. The closer J is to 1, the fairer throughput allocation is, and vices versa. To validate Adaptive-BBR's improvement of fairness, we conduct following extensive experiments. Each



(a) BBR (b) BBQ (c) Adaptive-BBR

Fig. 3. Adaptive-BBR makes two competing flows occupy similar share of the BtlBw, however BBR and BBQ obviously bias against 5-ms flow.

experiment is repeated at least three times and lasts for at least 60 seconds. The default bottleneck buffer size is 5 MB.

- **Bottleneck bandwidth**. 10-ms RTT flow competes with 15-ms RTT flow under different bottleneck bandwidth (10 Mbps, 20 Mbps and 50 Mbps). The results in Fig. 4(a) indicates that the index (around 0.6) of BBR is the smallest under different bottleneck bandwidth. BBQ can improve the index to a certain extent (around 0.84), but the effect is limited compared with our scheme. Among all the three algorithms, Adaptive-BBR obtains the largest Jain index (above 0.95) under different bottleneck bandwidth, which is always close to 1.
- **Disparity of RTT**. Under 20 Mbps bottleneck link bandwidth, 10-ms RTT flow competes with different RTT flow (15-ms, 20-ms, 30-ms and 50-ms). Fig. 4(b) shows BBR gets the smallest index, which is close to 0.5 (the worst) when the RTT is 30-ms. BBQ can obviously improve the index to around 0.9 when RTT is higher than 15 ms. However, the index of Adaptive-BBR is above 0.94 under different disparity of RTT, achieving the best fairness among three algorithms.
- **Number of competing flows**. One 10-ms RTT flow competes with different number of 15-ms RTT flows under 20 Mbps bottleneck bandwidth. Fig. 4(c) indicates that the share of the 10-ms BBR flow on the bandwidth is below 12%, and it only occupies 5.9% of bandwidth when competing with five 15-ms RTT flows. The share (28.6%) of 10-ms BBQ flow is around 2.6 times higher than that of BBR, when 10-ms RTT flow competes with only one 15-ms RTT flow. But, as the number of 15-ms RTT flow increases, the share of 10-ms BBQ flow gradually approaches that of BBR. When there are five 15-ms RTT flows, the share of 10-ms BBQ flow is only 6.6%, which is close to that of the 10-ms BBR flow's (5.9%). The shares of 10-ms Adaptive-BBR flow are respectively $53.8\%, 28.2\%$ and 19.4%, when the number of 15-ms RTT flows increases from 1 to 5, which implies that flows are fairly sharing the bottleneck bandwidth.

Adaptive-BBR's indexes given by Fig. 4(a) and Fig. 4(b) are all very close to 1, and its index is at least 47.8% higher than that of BBR. As the number of competing flows increases, Fig. 4(c) shows the gain of BBQ is vanishing. However, Adaptive-BBR keeps different flows occupy similar bandwidth.

C. *Latency*

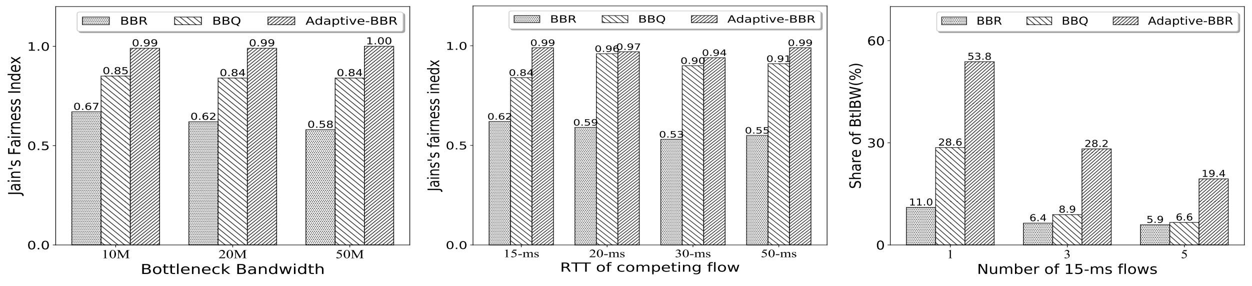
We consider 5-ms RTT flow competes with one 20-ms RTT flow under 20 Mbps bottleneck bandwidth. The bottleneck buffer size is 5 MB. Fig. 5 shows the behavior of RTT in three different schemes. Fig. 5(a) shows RTT of the 5-ms BBR flow increases to around 23 ms most of the time. Fig. 5(b) indicates 5-ms BBQ flow experiences around 13 ms. As Fig. 5(c) shows, Adaptive-BBR experiences smaller latency (around 6.2 ms) than the other two algorithms and it reduces average queuing delay by 93.3% compared with BBR (around 23 ms). The result given by Fig. 5(d) implies that RTT of 5-ms Adaptive-BBR flow during first 12 s and the latency roughly converges to 6.2 ms at around 1.5 s. In Fig. 5, the previous peak of RTT appears when large number of packets are inserted into the pipe during the startup phase. Fig. 5(a) also verifies the conclusion of Section III-B. Because aggregate sending rate mismatches bottleneck bandwidth when there are multiple flows, the created deep queue will lead to high latency until BBR enters ProbeRTT phase every 10 s. Although BBQ limits the number of packets inserted into the pipe by senders by using default duration of probing period, the latency it caused is still high.

D. *Responsiveness*

We consider bottleneck bandwidth of 20 Mbps capacity, 15-ms RTT flow joins and competes with 10-ms RTT flow at 3 s then it drops at 63 s. Fig. 6(a) shows BBR converges at around 18 s after 15-ms RTT flow joins. 10-ms RTT flow can quickly occupy available bandwidth at around 64 s after the others drop. As shown in Fig. 6(b), Adaptive-BBR converges at around 24 s after 15-ms RTT flow joins, which is slower than BBR. But 10-ms Adaptive-BBR flow can occupy the available bandwidth as quickly as 10-ms BBR flow, after 15-ms RTT flow quits. Similarly, before 15-ms RTT flow joins, 10-ms Adaptive-BBR flow obtains the bottleneck bandwidth as quickly as 10-ms BBR flow. Due to the adjustment scale of $\frac{\alpha - Q_i}{8}$ in every period, Adaptive-BBR converges more slowly than BBR when other flows join. But if there is available bandwidth, Adaptive-BBR can use 1.25 pacing gain to occupy available bandwidth as quickly as BBR.

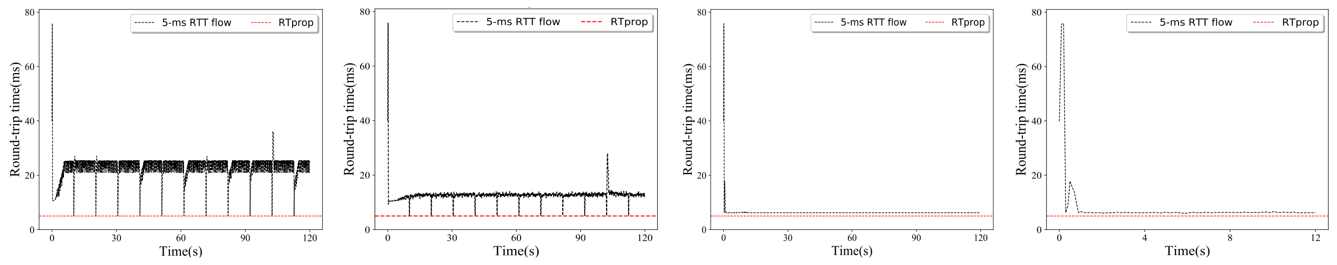
VI. Conclusion

In this paper, by analyzing the steady phase of BBR congestion control, we identified that, BBR flow's asynchronously



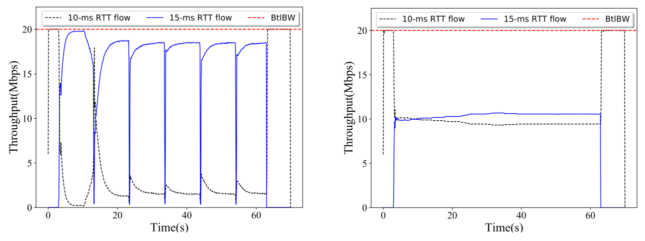
(a) Different BtlBw (b) Different disparity of RTT (c) Share of throughput

Fig. 4. Jain's Fairness Index and the share of throughput, under more network condition.



(a) BBR: RTT (b) BBQ: RTT (c) Adaptive-BBR: RTT (d) Adaptive-BBR: RTT (zoomed-in)

Fig. 5. 5-ms Adaptive-BBR flow experiences lower latency (around 6.2 ms) than BBR (23 ms) and BBQ (13 ms).



(a) BBR (b) Adaptive-BBR

Fig. 6. 15-ms flow joins at 3 s and drops at 63 s.

exploring of bottleneck bandwidth causes the mismatch between sending rate and bottleneck bandwidth, which is the root cause of unfairness and high latency. To address this problem, we proposed Adaptive-BBR, which controls sending rate via adaptive pacing gain. It guarantees that flows with different RTTs are fairly sharing the bottleneck bandwidth. Extensive experimental results show the proposed scheme outperforms BBR and BBQ with improved fairness and lower latency.

Acknowledgment

This work was supported by the National Natural Science Foundation of China under Grant 61871437 and Grant 61702205.

References

[1] S. Ha *et al.*, "CUBIC: a new TCP-friendly high-speed TCP variant," *ACM SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, pp. 64-74, Jul. 2008.

[2] B. Montazeri, Y. Li, M. Alizadeh, and J. Ousterhout, "Homa: A Receiver-Driven Low-Latency Transport Protocol Using Network Priorities, in *Proc. ACM SIGCOMM*, 2018, pp. 221-235.

[3] A. Narayan *et al.*, "Restructuring Endpoint Congestion Control, in *Proc. ACM SIGCOMM*, 2018, pp. 30-43.

[4] K. Fall, S. Floyd, "Simulation-based comparisons of Tahoe, Reno and SACK TCP," *ACM Comput. Commun. Rev.*, vol. 26 no. 3, pp. 5-21, Jul. 1996.

[5] N. Cardwell, Y. Cheng *et al.*, "BBR: Congestion-based Congestion Control," *Commun. ACM*, vol. 60, no. 2, pp. 58-66, Feb. 2017.

[6] P. Yang, N. Zhang *et al.*, "Catalyzing cloud-fog inter operation in 5g wireless networks: An SDN approach," *IEEE Netw.*, vol. 31, no. 5, pp. 14-20, Sept./Oct. 2017.

[7] P. Yang, N. Zhang *et al.*, "Content popularity prediction towards location-aware mobile edge caching," *IEEE Trans. Multimedia*, accepted, DOI: 10.1109/TMM.2018.2870521.

[8] M. Hock, R. Bless, and M. Zitterbart, "Experimental Evaluation of BBR Congestion Control," in *Proc. IEEE ICNP*, Oct. 2017, pp. 1-10.

[9] S. Ma, J. Jiang, W. Wang, and B. Li, "Fairness of Congestion-Based Congestion Control: Experimental Evaluation and Analysis," [Online]. arXiv: 1706.09115v2, 2017.

[10] Q. Peng, J. Hwang, and S. H. Low, "Multipath TCP: Analysis, Design, and Implementation," *IEEE/ACM Trans. Netw.*, vol. 24, no.1, pp. 596-609, Feb. 2016.

[11] R. Jain, D. Chiu, and W. Hawe, "Throughput fairness index: An explanation," ATM Forum Document: ATM Forum/99-0045, Feb. 1999.

[12] J. Gettys and K. Nichols, "Bufferbloat: Dark Buffers in the Internet," *ACM Queue*, vol. 9, no. 11, pp. 40-54, Nov. 2011.

[13] S. Hu, J. Li, and G. Pan, "Performance and Fairness Enhancement in IEEE 802.11 WLAN Networks," *AEU-Int J of Electron Commun.*, vol. 68, no. 7, pp. 667-675, Jul. 2014.

[14] V. N. Ha and L. B. Le, "Fair Resource Allocation for OFDMA Femtocell Networks With Macrocell Protection," *IEEE Trans. Veh. Technol.*, vol. 63, no. 3, pp. 1388-1401, Mar. 2014.

[15] I. K. Son, S. Mao, M. X. Gong, and Y. Li, "On Frame-based Scheduling for Directional mmWave WPANs," in *Proc. IEEE INFOCOM*, Orlando, FL, USA, Mar. 2012, pp. 2149-2157.

[16] L. X. Cai, L. Cai, X. Shen, and J. W. Mark, "REX: A Randomized Exclusive Region Based Scheduling Scheme for mmWave WPANs with Directional Antenna," *IEEE Trans. Wireless Commun.*, vol. 9, no. 1, pp. 113-121, Jan. 2010.

[17] G. Raina, S. Manjunath *et al.*, "Stability and Performance Analysis of Compound Tcp with Rem and Drop-tail Queue Management," *IEEE/ACM Trans. Netw.*, vol. 24, no. 4, pp. 1961-1974, Aug. 2016.

[18] K. Nichols and V. Jacobson, "Controlling Queue Delay," *ACM Queue*, vol. 55, no. 7, pp. 42-50, Jul. 2012.