Improving Inter-Protocol Fairness Based on Estimated Behavior of Competing Flows

Satoshi Utsumi
Cluster of Science and Technology
Fukushima University, Japan
u-satoshi@sss.fukushima-u.ac.jp

Go Hasegawa
Research Institute of Electrical Communication
Tohoku University, Japan
hasegawa@riec.tohoku.ac.jp

***Abstract*—Although the openness of the Internet is an excellent characteristic, the quality of service that users gain is not always fair due to the characteristic. Several versions of congestion control algorithms are used on the Internet. In recent years, CUBIC and BBR have taken a large share as congestion control algorithms, but throughput fairness among flows is not maintained when CUBIC and BBR flows compete in a bottleneck link. One possible method of improving throughput fairness is to regulate BBR flows based on the estimation of the average packet sending rate of competing CUBIC flows. Existing analytical models for estimating the performance of CUBIC flows cannot be used for improving throughput fairness because certain parameters in the models cannot be accurately obtained by competing flows. In this paper, we first present a novel analytical model to estimate the packet sending rate of competing CUBIC flows. We then propose a modification of BBR, named BBR-FCA, to improve throughput fairness with competing CUBIC flows. We evaluate the accuracy of our analytical model and performance of BBR-FCA through extensive simulations. The results indicate that the fairness index among CUBIC and BBR flows can be improved by up to 95.4%.**

***Index Terms*—CUBIC, BBR, Congestion control algorithm, Throughput fairness**

I. Introduction

The Internet is an open network and has been implemented in such a way that a variety of protocols can be used. This is an excellent characteristic of the Internet, but due to the characteristic, the quality of service that users gain is not always fair. For example, congestion control algorithms used by different operating systems (OSes) and content delivery network (CDN) operators at the transport and application layers are not the same [1], and their operational parameters are not also the same. When services with different congestion control algorithms share a bottleneck link, the quality of service may be inferior to that of other users. Conversely, the service may interfere with the quality of service of other users. In this paper, we focus on inter-protocol fairness, which is the fairness of the performance among different protocols.

As TCP congestion control algorithms, several versions are used in the Internet. The default TCP congestion control algorithm for Windows, macOS, Android, which are mainly used for end-user terminals, and Linux for servers is CUBIC [2], which is a loss-based congestion control algorithm. Bottleneck Bandwidth and Round-trip propagation time (BBR) [3], which is a congestion control algorithm for reducing the buffering delay at the bottleneck link, was developed by Google. It is used in Google's servers and certain CDN operator's servers [1]. The use of CUBIC and BBR in the current Internet is around 40 and 20%, respectively [4]. Therefore, there are more situations in which CUBIC and BBR flows compete on bottleneck links in the Internet, meaning that fairness among them is important. When BBR as a congestion control algorithm that reduces buffering delay competes with CUBIC as a loss-based congestion control algorithm, it is known that their throughput may be unfair, depending on network parameters such as bandwidth, buffer size, the number of flows in the bottleneck link, and round-trip time (RTT) [5], [6]. Especially BBR flows experience smaller or larger throughput than competing CUBIC flows, depending on the buffer size at the bottleneck link [5], [7]–[9]. The previous studies [10]–[17] present modifications of BBR for improving throughput fairness with CUBIC. However, their approaches are based on heuristic algorithms and they do not consider a quite large buffer at the bottleneck link, whereas much attention is now paid to the bufferbloat problem [18].

One possible direct method of improving throughput fairness among CUBIC and BBR flows is to regulate BBR flows based on the estimation of the average packet sending rate of competing CUBIC flows. There are several analytical models for estimating the performance of CUBIC flows [2], [19]–[24]. However, most cannot be used for improving throughput fairness because certain parameters required in the models, such as the frequency of packet loss detection events and packet loss ratio, cannot be accurately obtained by competing flows.

In this paper, we first present a novel analytical model to estimate the average packet sending rate of a CUBIC flow by competing flows. The model takes into account the congestion window dynamics of a CUBIC flow more precisely than existing models. We then propose BBR Fair to CUBIC based on mathematical Analysis (BBR-FCA), which is a modification of BBR based on the analytical model, to improve throughput fairness with competing CUBIC flows. Our modification is limited to control parameter settings of the original BBR to maintain its fundamental characteristics. The proposed method in this paper is not an incremental improvement based on a heuristic approach, but an essential

ISBN 978-3-903176-48-5

improvement based on mathematical analysis. We evaluate the accuracy of our analytical model and performance of BBR-FCA using a network simulator. We confirm that BBR-FCA can achieve high throughput fairness with CUBIC if the assumptions in our analytical model are held. For this purpose, we conduct extensive experiments changing network parameters.

The remainder of this paper is organized as follows. Section II summarizes previous works related to this study. In Section III, we briefly explain the CUBIC congestion control algorithm. In Section IV, we present our analytical model to estimate the average packet sending rate of a CUBIC flow. In Section V, we present our proposed BBR-FCA. We discuss the evaluation of the accuracy of the analytical model and that of the performance of BBR-FCA through simulations in Section VI. Finally, we conclude this paper and mention future works in Section VII.

II. Related Works

Several improvements for BBR have been proposed for the purpose of improving throughput fairness with loss-based congestion control algorithms such as CUBIC. The authors of [15] proposed a method to improve the fairness with CUBIC flows and excessive packet losses in a bottleneck link with small buffer size by updating the congestion window size of BBR flows when packet loss is detected, similar to BIC TCP [25]. The studies [16], [17] focused on the fact that when BBR flows compete with flows using loss-based congestion control algorithms in a bottleneck link, the round-trip propagation delay cannot be accurately observed as the minimum RTT. They proposed an improvement for the packet loss recovery of BBR to reduce the unfairness with loss-based congestion control algorithms. The proposals are avoiding excessive packet losses in a bottleneck link with small buffer size. In addition to the methods in [16], [17], the authors of [13] improved the congestion window control after a packet loss detection of BBR flows so that the window size does not become too large, when the buffer size of the bottleneck link is small. BBRv2 [10] has been proposed to improve the shortcomings of BBR, such as unfairness with loss-based congestion control algorithms and excessive packet losses in networks with small buffer size, by controlling the aggressiveness of packet transmission in the bandwidth probe while observing packet losses, and by modifying the minimum RTT observation algorithm. BBRv2+ [14] improved the responsiveness to dynamic changes in the available bandwidth and durability against random losses in BBRv2 by carefully adjusting the aggressiveness of packet transmission in the bandwidth probe while observing the increase or decrease in RTT. BBRv2+ has been designed to improve the performance characteristics of delay-based congestion control algorithms, such as inferior throughput when competing with loss-based congestion control algorithms in the network with large buffer size. By carefully observing the latest minimum RTT, BBRv2+ can determine whether it competes with loss-based congestion control algorithms, and if so, BBRv2+ can perform the bandwidth probe similar to BBRv2. In [11], a machine learning algorithm is used to determine whether a BBR flow competes with a CUBIC flow, and if so, the minimum RTT observation algorithm similar to BBRv2 is applied to improve throughput fairness with the CUBIC flow. All of the approaches in the studies [10], [14]–[17] try to achieve throughput fairness with CUBIC flows based on heuristic algorithms. In addition, these approaches do not assume the bottleneck link with quite large buffer size, while the continuous extremely large buffering delay, called *bufferbloat*, has recently attracted much attention. Therefore, in this paper, we modify the BBR algorithm for the purpose of improving throughput fairness with CUBIC flows in the same bottleneck link with any buffer size, using an approach based on mathematical analysis.

III. CUBIC Congestion Control Algorithm

In this paper, we discuss throughput fairness among long-lived CUBIC and BBR flows. Therefore, we focus on the congestion avoidance phase in the congestion control algorithms.

The congestion window size in the j-th congestion avoidance phase of a CUBIC flow is mainly determined from the CUBIC window growth function $W_{\mathrm{cubic},j}(t)$ [packets] in Eq. (1):

$$W_{\mathrm{cubic},j}(t) = C\left(t - K_j\right)^3 + W_{\max,j} \tag{1}$$

where t [sec] is the elapsed time from the beginning of the j-th congestion avoidance phase, and C is the scaling factor. $W_{\max,j}$ [packets] is determined using Eq. (2):

$$W_{\max,j} = \begin{cases} \frac{2-\beta}{2} W_{0,(j-1)} & W_{0,(j-1)} < W_{\max,(j-1)} \quad (2.1) \\ W_{0,(j-1)} & \text{otherwise} \quad (2.2) \end{cases} \tag{2}$$

where $W_{0,(j-1)}$ [packets] is the congestion window size when a packet loss detection event occurs in the $(j-1)$-th congestion avoidance phase. K_j [sec] is the time it takes for the congestion window size to reach $W_{\max,j}$ [packets] from the beginning of the j-th congestion avoidance phase, determined using Eq. (3):

$$K_j = \sqrt[3]{\frac{W_{\max,j} - (1-\beta) W_{0,(j-1)}}{C}} \tag{3}$$

where β is the congestion window reduction ratio.

IV. Analytical Model for Estimating Packet Sending Rate of CUBIC Flow

The purpose of the analysis is to estimate the average packet sending rate of a CUBIC flow *by competing flows*. We assume that these flows share only one bottleneck link and that packet losses are due only to buffer overflows at the bottleneck link.

We develop our analytical model for the packet sending rate of a CUBIC flow with respect to the duration of the congestion avoidance phase, not to the frequency of packet loss detection events (i.e., congestion window halving rate [26]) or packet loss ratio as in the most studies [2], [19]–[23]. This is

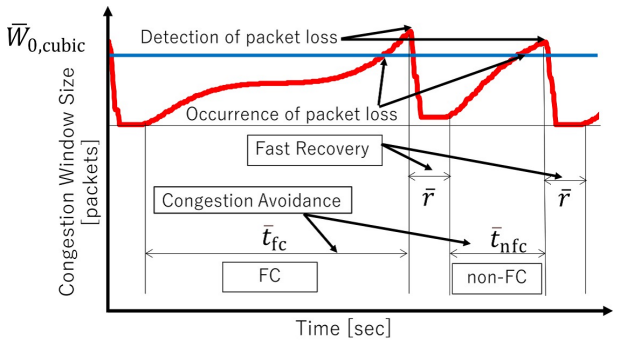

Fig. 1: Actual congestion window dynamics.

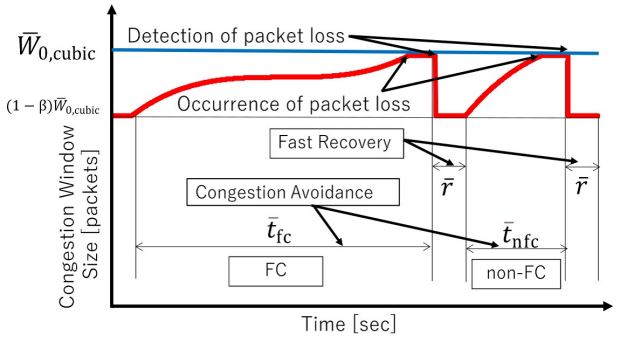

Fig. 2: Congestion window dynamics in our analysis.

because the packet sending rate cannot be accurately obtained from the frequency of packet loss detection events in the mixture of CUBIC and competing flows. Another reason is that competing flows cannot accurately obtain the frequency of the packet loss detection events on the CUBIC flow. In our work, by assuming that all co-existing flows on the bottleneck link have packet losses simultaneously when a buffer overflow occurs, the duration of congestion avoidance phases of the CUBIC flow should be estimated easily by competing flows.

Eq. (2) means that the evolution of the congestion window size of a CUBIC flow has two modes. Therefore, we give the mathematical analysis for both modes. The congestion avoidance phase with Eq. (2.1) is called the Fast Convergence (FC) phase, and that with Eq. (2.2) is called the non-FC phase. We construct our analytical model based on the model in [24]. In [24], the authors derived the average congestion window size and duration of the congestion avoidance phase of a CUBIC flow as a function of the frequency of the packet loss detection events. The main difference is that we precisely consider the effect of FC and non-FC phases. Another difference is that we explicitly consider the effect of the fast recovery phase as well as the congestion avoidance phase.

Fig. 1 shows the typical dynamics of the congestion window size in the congestion avoidance phases of a CUBIC flow obtained from a simulation. We observe the repetition of a congestion avoidance phase and following fast recovery phase. Each congestion avoidance phase is either an FC phase or non-FC phase. We denote the average duration of an FC phase and that of a non-FC phase as $\bar{t}_{\mathrm{fc}}$ [sec] and $\bar{t}_{\mathrm{nfc}}$ [sec], respectively. We also denote the average duration of a fast recovery phase as $\bar{r}$ [sec]. The average congestion window size when the packet is discarded in the network is denoted as $\overline{W}_{0,\mathrm{cubic}}$ [packets].

Both FC and non-FC phases terminate when packet losses are detected. Due to the packet loss detection algorithm, it takes roughly one RTT to detect the packet loss. The evolution of the congestion window size in the duration between the packet loss and its detection depends on various factors, such as how many packets are discarded. Therefore, we ignore the congestion window evolution in the duration between a packet loss and its detection, and assume that the congestion window size remains $\overline{W}_{0,\mathrm{cubic}}$. We also ignore the evolution of the congestion window size in the fast recovery phase and fix it as $(1-\beta)\overline{W}_{0,\mathrm{cubic}}$ for simplification of the analysis. Fig. 2 illustrates the evolution of the congestion window size by reflecting these assumptions. We confirmed that they do not affect the overall estimation accuracy of the analysis.

We first derive the average congestion window size in a congestion avoidance phase, denoted as $\overline{w}_{\mathrm{cubic}}$ [packets]. The average congestion window evolution from the beginning of the congestion avoidance phase to when packet loss occurs in FC and non-FC phases, denoted as $\overline{W}_{\mathrm{fc}}(t)$ [packets] and $\overline{W}_{\mathrm{nfc}}(t)$ [packets] respectively, is approximated as follows, based on Eqs. (1)-(3) and the above assumptions.

$$\overline{W}_{\mathrm{fc}}(t) = C\left(t - \sqrt[3]{\frac{\beta\overline{W}_{0,\mathrm{cubic}}}{2C}}\right)^3 + \frac{2-\beta}{2}\overline{W}_{0,\mathrm{cubic}} \quad (4)$$

$$\overline{W}_{\mathrm{nfc}}(t) = C\left(t - \sqrt[3]{\frac{\beta\overline{W}_{0,\mathrm{cubic}}}{C}}\right)^3 + \overline{W}_{0,\mathrm{cubic}} \quad (5)$$

We denote the average RTT as $\overline{R}_{\mathrm{cubic}}$ [sec]. Then, $\bar{t}_{\mathrm{fc}}$ and $\bar{t}_{\mathrm{nfc}}$ are obtained as follows, by substituting $(\bar{t}_{\mathrm{fc}} - \overline{R}_{\mathrm{cubic}})$ into t in Eq. (4) and $(\bar{t}_{\mathrm{nfc}} - \overline{R}_{\mathrm{cubic}})$ into t in Eq. (5), respectively.

$$\bar{t}_{\mathrm{fc}} = \sqrt[3]{\frac{4\beta\overline{W}_{0,\mathrm{cubic}}}{C}} + \overline{R}_{\mathrm{cubic}} \quad (6)$$

$$\bar{t}_{\mathrm{nfc}} = \sqrt[3]{\frac{\beta\overline{W}_{0,\mathrm{cubic}}}{C}} + \overline{R}_{\mathrm{cubic}}$$

From Eq. (6), we obtain the following equation.

$$\overline{W}_{0,\mathrm{cubic}} = \frac{C}{4\beta}(\bar{t}_{\mathrm{fc}} - \overline{R}_{\mathrm{cubic}})^3$$

We next derive the average congestion window size from the beginning of the congestion avoidance phase to when packet loss occurs in FC and non-FC phases, denoted as

$\overline{w}_{\rm fc}$ [packets] and $\overline{w}_{\rm nfc}$ [packets], respectively, in accordance with the following calculations.

$$\begin{aligned}\overline{w}_{\rm fc} &= \frac{1}{\bar{t}_{\rm fc}-\overline{R}_{\rm cubic}}\int_0^{\bar{t}_{\rm fc}-\overline{R}_{\rm cubic}}\overline{W}_{\rm fc}(t)dt \\ &= \frac{2-\beta}{8\beta}C(\bar{t}_{\rm fc}-\overline{R}_{\rm cubic})^3\end{aligned}$$

$$\begin{aligned}\overline{w}_{\rm nfc} &= \frac{1}{\bar{t}_{\rm nfc}-\overline{R}_{\rm cubic}}\int_0^{\bar{t}_{\rm nfc}-\overline{R}_{\rm cubic}}\overline{W}_{\rm nfc}(t)dt \\ &= \frac{4-\beta}{16\beta}C(\bar{t}_{\rm fc}-\overline{R}_{\rm cubic})^3\end{aligned}$$

We introduce the parameter γ as the ratio of the number of FC phases relative to the total number of congestion avoidance phases. Then, $\overline{w}_{\rm cubic}$ can be obtained by averaging the congestion window size in a congestion avoidance phase as follows:

$$\overline{w}_{\rm cubic} = \frac{\gamma(\bar{t}_{\rm fc}-\overline{R}_{\rm cubic})\overline{w}_{\rm fc} + (1-\gamma)(\bar{t}_{\rm nfc}-\overline{R}_{\rm cubic})\overline{w}_{\rm nfc} + \overline{R}_{\rm cubic}\overline{W}_{0,\rm cubic}}{\bar{t}} \tag{7}$$

where $\bar{t}$ is the average duration of the congestion avoidance phase calculated as the following equation.

$$\bar{t} = \gamma\bar{t}_{\rm fc} + (1-\gamma)\bar{t}_{\rm nfc}$$

Note that the term $\overline{R}_{\rm cubic}\overline{W}_{0,\rm cubic}$ in Eq. (7) is for the duration from when packet losses occur to when they are detected. The average packet sending rate in a congestion avoidance phase can then be obtained as $\frac{\overline{w}_{\rm cubic}}{\overline{R}_{\rm cubic}}$ [packets/sec].

We next calculate the average packet sending rate in a fast recovery phase by dividing the number of packets sent in this phase by the duration of the phase. We assume that the number of packets sent in this phase is $(1-\beta)\overline{W}_{0,\rm cubic}$, which equals the congestion window size in the phase, on the basis of the observations in simulations. Therefore, the average packet sending rate in a fast recovery phase can be obtained as $\frac{(1-\beta)\overline{W}_{0,\rm cubic}}{\bar{r}}$ [packets/sec].

Finally, we obtain the average packet sending rate of a CUBIC flow, denoted as $\overline{B}_{\rm cubic}$ [packets/sec], as follows.

$$\overline{B}_{\rm cubic} = \frac{\bar{t}\frac{\overline{w}_{\rm cubic}}{\overline{R}_{\rm cubic}} + \bar{r}\frac{(1-\beta)\overline{W}_{0,\rm cubic}}{\bar{r}}}{\bar{t}+\bar{r}} \tag{8}$$

CUBIC ensures that the packet sending rate is kept equal or larger than that of TCP Reno [27], which is the classic congestion control algorithm used mainly on the Internet in the past. This is possible by estimating the congestion window size of a TCP Reno flow passing through the same network path. We take this behavior into account in our analysis, but the details are omitted due to space limitation.

V. BBR-FCA: BBR Fair to CUBIC Based on Mathematical Analysis

In this section, we propose a modification of the BBR congestion control algorithm to improve throughput fairness with CUBIC, named BBR-FCA.

For simplification of the explanation, we consider the situation in which one CUBIC flow and one BBR-FCA flow compete on a single bottleneck link. The BBR-FCA flow regulates its packet sending rate to maintain throughput fairness with the competing CUBIC flow. To do this, it first estimates the packet sending rate of the CUBIC flow using the analysis results presented in Section IV. We need to set parameters C, β, $\overline{R}_{\rm cubic}$, $\bar{r}$, γ, $\bar{t}_{\rm fc}$, and $\bar{t}_{\rm nfc}$ to estimate the packet sending rate of the CUBIC flow using Eq. (8). We discuss how these parameters are determined in Section V-A. In Section V-B, we give an overview of the original BBR algorithm. Finally, in Section V-C, we describe how to configure the parameters of BBR-FCA and details of this algorithm.

A. Parameter Settings of CUBIC Analytical Model

We use $C = 0.4$ and $\beta = 0.3$, which are used in Linux and Android. For $\overline{R}_{\rm cubic}$, we use the smoothed RTT maintained by the BBR-FCA flow, which is denoted as $\overline{R}_{\rm bbr}$ [sec].

As in Section IV, we assume that when a buffer overflow occurs at a bottleneck link, packet losses occur in both CUBIC and BBR-FCA flows simultaneously. The BBR-FCA flow can then estimate the duration of the congestion avoidance phases of the CUBIC flow by observing the packet loss detection events and packet loss recoveries. It can also determine $\bar{r}$, by estimating two successive congestion avoidance phases. The problem is that we cannot determine whether the observed congestion avoidance phase corresponds to the FC phase or non-FC phase. Therefore, the BBR-FCA flow estimates $\bar{t}_{\rm fc}$ and $\bar{t}_{\rm nfc}$ with the following equations:

$$\bar{t}_{\rm fc} = t_{\rm avg} + \sqrt{t_{\rm var}}$$

$$\bar{t}_{\rm nfc} = t_{\rm avg} - \sqrt{t_{\rm var}}$$

where $t_{\rm avg}$ and $t_{\rm var}$ are the moving average and variance of the duration of the congestion avoidance phases, respectively. This estimation is based on the following two assumptions. First, we assume that an FC phase and non-FC phase appear alternately, meaning that γ equals 0.5. Second, all FC phases have longer duration than the average duration of a congestion avoidance phase, and all non-FC phases have shorter duration than the average duration of a congestion avoidance phase. Note that we empirically confirmed that these assumptions are satisfied in the simulations we conducted for this study.

B. Original BBR Algorithm [3]

In the long-lived BBR flow, the ProbeBW and ProbeRTT phases appear alternately. In the ProbeBW phase, the BBR flow sets the target value for the congestion window size, and the congestion window size gradually approaches the target value. The target value at receiving the i-th ACK packet is denoted as $w_{{\rm target},i}$ [packets]. In addition, the BBR flow executes packet pacing when sending packets. The pacing rate is denoted as p_i [packets/sec]. To determine $w_{{\rm target},i}$ and p_i, the BBR flow uses the maximum throughput, denoted as $BtlBw_i$ [packets/sec], calculated from the reception of ACK packets in the recent past, and the minimum RTT in the recent past, denoted as $RT_{{\rm prop},i}$ [sec]. Time windows for calculating

$BtlBW_i$ and $RT_{\mathrm{prop},i}$, denoted as W_{B} [sec] and W_{R} [sec], are set to ten times RTT and 10 [sec], respectively, as in the implementation of Linux 5.14. The $w_{\mathrm{target},i}$ and p_i are calculated as follows:

$$w_{\mathrm{target},i} = g_{\mathrm{w}} \cdot BtlBw_i \cdot RT_{\mathrm{prop},i}$$
$$p_i = g_{\mathrm{p}} \cdot BtlBw_i \tag{9}$$

where g_{w} and g_{p} are the fixed parameters called the cwnd gain and pacing gain, respectively. g_{p} is temporarily changed to search for the available bandwidth in the ProbeBW phase.

In the ProbeRTT phase, the congestion window size is set to δ [packets] and new packets are sent during W_{rtt} [sec]. $\delta = 4$ [packets] and $W_{\mathrm{rtt}} = 0.2$ [sec] are used in the implementation of BBR in Linux 5.14.

C. BBR-FCA Algorithm

Parameter Configuration: In BBR-FCA, we modify the algorithms to determine the target value for the congestion window size, packet pacing rate and cwnd gain.

The average packet sending rate of the original BBR flow, $\overline{S}$ [packets/sec], is expressed as Eq. (10):

$$\overline{S} = \frac{\frac{\delta+\overline{w}_{\mathrm{target}}}{2\overline{R}_{\mathrm{bbr}}}\bar{t}_{\mathrm{bw}} + \frac{\overline{w}_{\mathrm{target}}}{\overline{R}_{\mathrm{bbr}}}(W_{\mathrm{R}} - \bar{t}_{\mathrm{bw}}) + \frac{\delta}{\overline{R}_{\mathrm{bbr}}}W_{\mathrm{rtt}}}{W_{\mathrm{R}} + \bar{t}_{\mathrm{rtt}}} \tag{10}$$

where $\overline{w}_{\mathrm{target}}$ [packets] is the average target value for the congestion window size, $\bar{t}_{\mathrm{bw}}$ [sec] is the average time from the beginning of the ProbeBW phase to when the congestion window size reaches $\overline{w}_{\mathrm{target}}$, and $\bar{t}_{\mathrm{rtt}}$ [sec] is the average duration of the ProbeRTT phase. Eq. (10) can be obtained by deriving the weighted average of the congestion window sizes in the ProbeBW and ProbeRTT phases. The details of the derivation of Eq. (10) is omitted due to space limitation. By solving Eq. (10) for $\overline{w}_{\mathrm{target}}$, we obtain Eq. (11).

$$\overline{w}_{\mathrm{target}} = \frac{\overline{S}\ \overline{R}_{\mathrm{bbr}}(W_{\mathrm{R}} + \bar{t}_{\mathrm{rtt}}) - \delta(W_{\mathrm{rtt}} + \frac{\bar{t}_{\mathrm{bw}}}{2})}{W_{\mathrm{R}} - \frac{\bar{t}_{\mathrm{bw}}}{2}} \tag{11}$$

Eq. (11) means that we can obtain the average target value for the congestion window size so that the average packet sending rate of the BBR flow can be $\overline{S}$. By substituting $\overline{B}_{\mathrm{cubic}}$ in Eq. (8), calculated from the estimated CUBIC parameters, into $\overline{S}$ in Eq. (11), the packet sending rate of the BBR flow can be regulated to maintain throughput fairness with the competing CUBIC flow.

We also modify the packet pacing rate, by replacing $BtlBW_i$ in Eq. (9) with $\overline{p}$ [packets/sec] in Eq. (12).

$$\overline{p} = \frac{\overline{w}_{target}}{\overline{R}_{\mathrm{bbr}}} = \frac{\overline{S}\ \overline{R}_{\mathrm{bbr}}(W_{\mathrm{R}} + \bar{t}_{\mathrm{rtt}}) - \delta(W_{\mathrm{rtt}} + \frac{\bar{t}_{\mathrm{bw}}}{2})}{\overline{R}_{\mathrm{bbr}}(W_{\mathrm{R}} - \frac{\bar{t}_{\mathrm{bw}}}{2})} \tag{12}$$

Finally, we adaptively configure the cwnd gain g_{w} depending on $\overline{w}_{\mathrm{target}}$, $\overline{p}$ and $RT_{\mathrm{prop},i}$, as in Eq. (13).

$$g_{\mathrm{w}} = \frac{\overline{w}_{\mathrm{target}}}{\overline{p} \cdot RT_{\mathrm{prop},i}} \tag{13}$$

Details of BBR-FCA Algorithm: A BBR-FCA flow determines that CUBIC flows compete on a bottleneck link when packet losses are detected in the recent past. It then modifies its behavior in the ProbeBW phase as follows.

The BBR-FCA flow first estimates $\overline{B}_{\mathrm{cubic}}$ by using Eq. (8) with parameter values determined in Section V-A. By substituting $\overline{B}_{\mathrm{cubic}}$ into $\overline{S}$ in Eqs. (11) and (12), it then calculates $\overline{w}_{\mathrm{target}}$ and $\overline{p}$, respectively. Also, g_{w} is obtained from Eq. (13).

The original BBR flow temporarily changes g_{p} to search for the available bandwidth in ProbeBW phases. The BBR-FCA flow does not need to search for the available bandwidth when it competes with CUBIC flows, when focusing on achieving throughput fairness with these flows. Therefore, we omit the temporal changes of g_{p}. That is, $g_{\mathrm{p}} = 1$ constant.

In Startup, Drain and ProbeRTT phases, the BBR-FCA flow behaves the same to the original BBR flow.

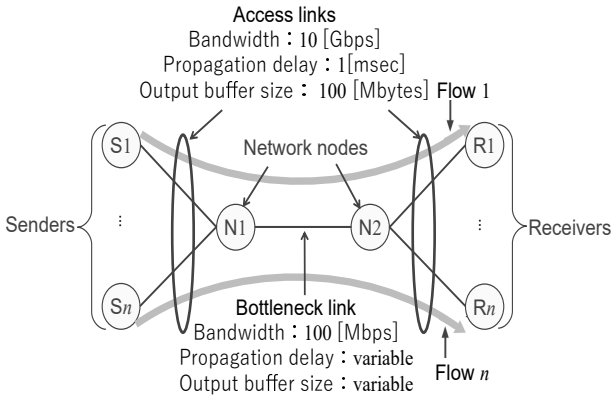

Fig. 3: Network topology.

VI. Evaluation

In this section, we validate the analytical model presented in Section IV, and evaluate throughput fairness among CUBIC and BBR flows through simulations. We use ns-2 [28] for these simulations since ns-2 has sufficient functionality to evaluate the performance of CUBIC and BBR.

A. Experimental Environment

Fig. 3 shows the network topology used in the simulations. The network consists of n senders, n receivers, two network nodes, $2n$ access links and one bottleneck link.

As mentioned in Section I, CUBIC and BBR are the dominant congestion control algorithms used in the current Internet environment. Therefore, only CUBIC and BBR are considered as TCP congestion control algorithms in the performance evaluation of this section.

Each sender is equipped with CUBIC, the original BBR, BBR-CWS [13], BBRv2 [10], BBRv2+ [14] or BBR-FCA. The six algorithms are implemented based on the source codes for Linux kernel, the literature [2], [3], [10], [13], [14] and Section V-C. Senders enable TCP pacing function. Receivers enable selective ACK option and disable delayed ACK option. The data and ACK packet sizes are 1500 and 40 [bytes], respectively. The bandwidth of the bottleneck link is 100 [Mbps], unless we mention otherwise. The propagation

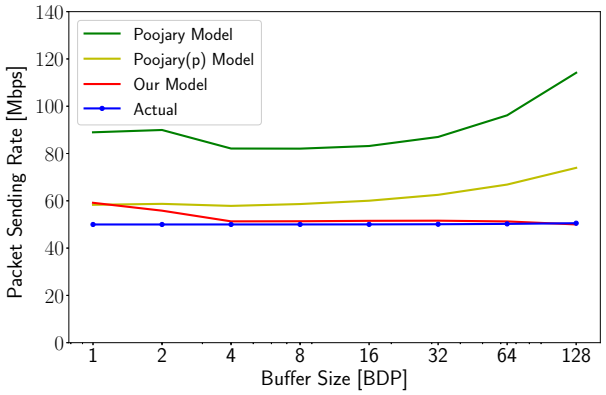

Fig. 4: Validation of analytical model: two CUBIC flows (propagation delay between terminals: 20 [msec]).

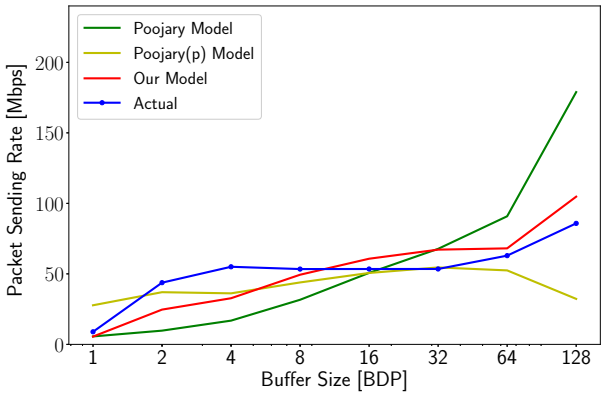

Fig. 5: Validation of analytical model: one CUBIC flow and one BBR flow (propagation delay between terminals: 20 [msec]).

delay of the bottleneck link is 18 or 38 [msec]. The output buffer size of the bottleneck link is 1, 2, 4, 8, 16, 32, 64 or 128 [BDP], where BDP represents the bandwidth delay product between the senders and receivers. 1 [BDP] corresponds to 500 [Kbytes] and 1 [Mbytes] in the network where the propagation delay of the bottleneck link is 18 and 38 [msec], respectively. Note that we conduct the performance evaluation with large buffer sizes by taking into account the bufferbloat problem. The bandwidth, propagation delay and output buffer size of access links are 10 [Gbps], 1 [msec] and 100 [Mbytes], respectively. Long-lived data transfers are started simultaneously from all senders. The duration of the data transfer is 6000 [sec].

B. Validation for Analytical Model

We use the two analytical models based on [24] as comparison models. The Poojary model estimates the average packet sending rate from the average duration of the congestion avoidance phase and the average congestion window size. The Poojary(p) model estimates the average packet sending rate from the frequency of packet loss detection events and the average congestion window size.

Fig. 4 shows the estimation results (Our Model, Poojary Model and Poojary(p) Model) regarding the average packet sending rate of a CUBIC flow as well as the actual values obtained from simulations (Actual), as a function of the buffer size, when two CUBIC flows exist in the network where the propagation delay of the bottleneck link is 18 [msec], that is, the propagation delay between terminals is 20 [msec]. Our model accurately estimates the packet sending rate, except for 1 and 2 [BDP]. This measurement error is due to the TCP pacing function enabled in the simulations, which is not considered in our analysis. The Poojary and Poojary(p) models, which do not take into account FC and non-FC phases, overestimate the average actual rate. The Poojary model overestimates more than the Poojary(p) model because of the difference in the effect of ignoring FC and non-FC phases on the analysis results.

Fig. 5 shows the results when one CUBIC flow and one BBR flow co-exist in the network. The original BBR is used for the BBR flow. This estimation is conducted by the BBR flow. Our model underestimates the packet sending rate when the buffer size is small, while providing almost accurate results for larger buffer sizes. This is because successive packet losses occur when the buffer size is small due to the BBR algorithm on the packet loss detection, which is not considered in our analysis. The Poojary model underestimates and overestimates the packet sending rate when the buffer size is small and large, respectively, because it does not take into account the difference in FC and non-FC phases. The Poojary(p) model underestimates the packet sending rate compared with the actual values when the buffer size is quite large, because the frequency of the packet loss detection events of the CUBIC flow cannot be accurately obtained by the BBR flow.

C. Throughput Fairness among CUBIC and BBR Flows

Here, we confirm whether BBR-FCA can achieve high throughput fairness with CUBIC or not, when the assumptions in Section IV are held. For this purpose, we conduct extensive experiments in this section.

We then evaluate throughput fairness among CUBIC and BBR (the original BBR, BBR-CWS, BBRv2, BBRv2+ or BBR-FCA) flows. Fig. 6 shows the throughput share of two flows as a function of the buffer size when a CUBIC flow and BBR flow co-exist in the network where the propagation delay of the bottleneck link is 18 [msec]. When the buffer size is quite small or quite large, the fairness between the CUBIC flow and the original BBR, or BBR-CWS flow degrades significantly. When the buffer size is quite large, the fairness between the CUBIC flow and the BBRv2 flow also degrades. When the buffer size is quite small, the fairness between the CUBIC flow and the BBRv2+ flow also degrades significantly. On the other hand, the BBR-FCA flow realizes almost perfect fairness regardless of the buffer size while maintaining full utilization of the bottleneck link bandwidth. These results clearly indicate the fundamental characteristics of BBR-FCA.

Fig. 7 shows the results when one CUBIC flow and m (≥ 1) BBR flows co-exist in the network where the propagation delay of the bottleneck link is 18 [msec] and the buffer size is set to 64 [BDP]. The graph shows Jain's Fairness Index [29],

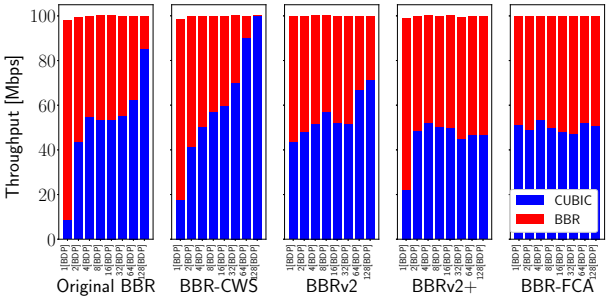

Fig. 6: Throughput share of one CUBIC flow and one BBR flow (propagation delay between terminals: 20 [msec]).

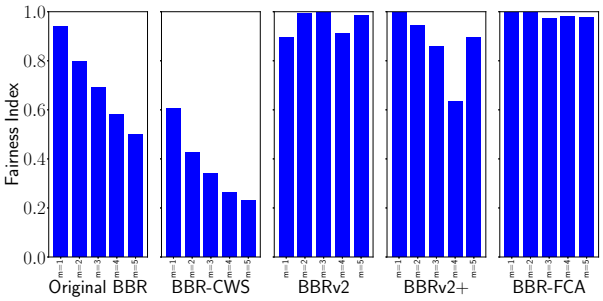

Fig. 7: Fairness index among one CUBIC flow and multiple (m) BBR flows (propagation delay between terminals: 20 [msec], buffer size: 64 [BDP]).

which is calculated from the throughput of the CUBIC flow and the throughput of each BBR flow. The fairness index takes a value between 0 and 1.0. The value is closer to 1.0, it indicates fairer. When the original BBR is used for the BBR flows, the fairness index decreases as the number of BBR flows increases; when BBRv2, BBRv2+, or BBR-FCA is used, the fairness index among those flows is greatly improved regardless of the number of BBR flows. Specifically, BBRv2 improves the fairness index by -4.6% to 96.5%, BBRv2+ improves it by 6.1% to 79.3%, and BBR-FCA improves it by 6.2% to 95.4%, comparing with the original BBR.

Fig. 8 shows the results when m (≥ 1) CUBIC flows and one BBR flow co-exist in the network where the propagation delay of the bottleneck link is 18 [msec] and the buffer size is set to 64 [BDP]. The graph shows Jain's Fairness Index, which is calculated from the throughput of each CUBIC flow and the throughput of the BBR flow. When the original BBR is used for the BBR flows, the fairness index decreases as the number of BBR flows increases; when BBR-FCA is used, the fairness index among those flows is improved. Specifically, BBR-FCA improves the fairness index by -2.2% to 27.9%, comparing with the original BBR.

Fig. 9 shows the Jain's Fairness Index of multiple CUBIC and BBR flows when the same number ($= m$) of such flows co-exist in the network where the propagation delay of the bottleneck link is 18 [msec] and the buffer size is set to 64 [BDP]. With all of the BBR-CWS, BBRv2, and BBRv2+, the fairness decreases when the number of flows is large. On

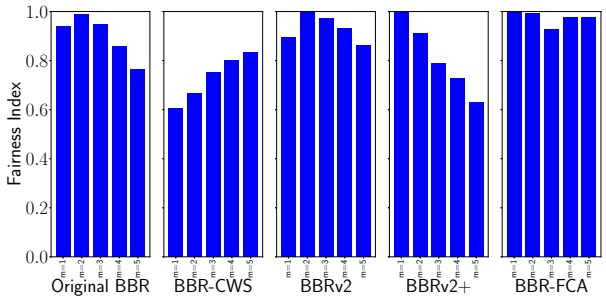

Fig. 8: Fairness index among multiple (m) CUBIC flows and one BBR flow (propagation delay between terminals: 20 [msec], buffer size: 64 [BDP]).

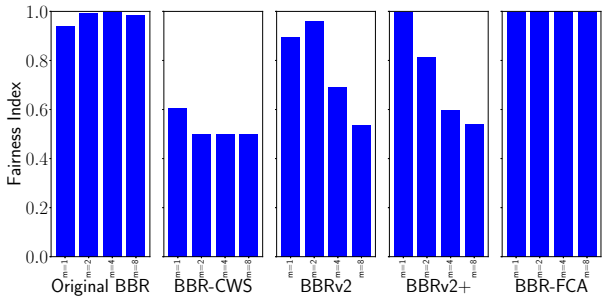

Fig. 9: Fairness index among multiple (m) CUBIC flows and multiple (m) BBR flows (propagation delay between terminals: 20 [msec], buffer size: 64 [BDP]).

the other hand, with BBR-FCA, the fairness is close to 1.0 regardless the number of flows. These results indicate that BBR-FCA can be used in a situation with multiple CUBIC and BBR flows, whereas the analysis in Section IV implicitly assumes that one CUBIC flow and one BBR flow share a bottleneck link.

Next, we change the propagation delay and the bandwidth of the bottleneck link to gain extensive experimental results.

Fig. 10 shows the throughput share of two flows as a function of the buffer size when a CUBIC flow and BBR flow co-exist in the network where the propagation delay of the bottleneck link is 38 [msec], that is, the propagation delay between terminals is 40 [msec]. The original BBR, BBR-CWS and BBRv2 have the results similar to when the propagation delay of the bottleneck link is 18 [msec]. The BBRv2+ flow can achieve throughput which is fair to the CUBIC flow even when the buffer size of the bottleneck link is 1 [BDP], because the buffer size is larger than when the propagation delay of the bottleneck link is 18 [msec]. On the other hand, when the buffer size is 128 [BDP] (= 128 [Mbytes]), the BBR-FCA flow cannot achieve throughput fairness with the competing CUBIC flow. The reason is because retransmission timeouts are frequent in this environment. This frequent timeouts are due to quite large buffer size rather than due to large propagation delay. Our analysis does not take a retransmission timeout into account. Therefore, the BBR-FCA flow is not fair to the competing CUBIC flow in the environment where retransmission timeouts are frequent, such

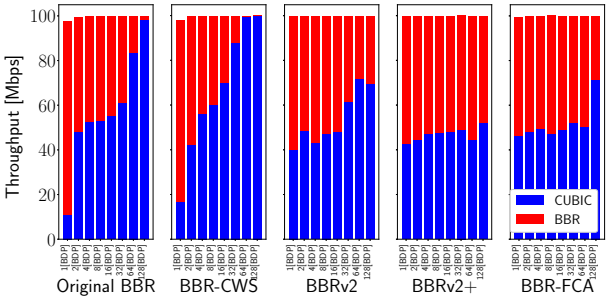

Fig. 10: Throughput share of one CUBIC flow and one BBR flow (propagation delay between terminals: 40 [msec]).

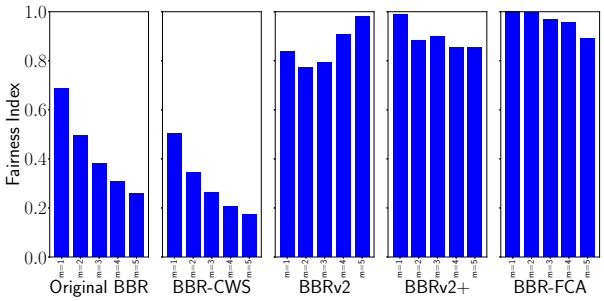

Fig. 11: Fairness index among one CUBIC flow and multiple (m) BBR flows (propagation delay between terminals: 40 [msec], buffer size: 64 [BDP]).

as when the buffer size of the bottleneck link is quite large. Basically, the magnitude of the propagation delay between terminals has little effect on the throughput fairness among CUBIC flows and BBR-FCA flows.

Fig. 11 shows the results when one CUBIC flow and m ($\geq$ 1) BBR flows co-exist in the network where the propagation delay of the bottleneck link is 38 [msec] and the buffer size is set to 64 [BDP]. The graph shows Jain's Fairness Index, which is calculated from the throughput of the CUBIC flow and the throughput of each BBR flow. The original BBR, BBR-CWS, BBRv2, BBRv2+ and BBR-FCA have worse fairness a little than when the propagation delay of the bottleneck link is 18 [msec]. When BBRv2, BBRv2+, or BBR-FCA is used, the fairness index among those flows is greatly improved regardless of the number of BBR flows, compared with the original BBR. Specifically, BBRv2 improves the fairness index by 22.0% to 281.4%, BBRv2+ improves it by 44.0% to 232.9%, and BBR-FCA improves it by 45.6% to 247.0%, comparing with the original BBR.

Fig. 12 shows the results when m ($\geq$ 1) CUBIC flows and one BBR flow co-exist in the network where the propagation delay of the bottleneck link is 38 [msec] and the buffer size is set to 64 [BDP]. The graph shows Jain's Fairness Index, which is calculated from the throughput of each CUBIC flow and the throughput of the BBR flow. BBR-CWS, BBRv2, BBRv2+ and BBR-FCA have the results similar to Fig. 8. When the original BBR is used for the BBR flows, the fairness index decreases as the number of BBR flows decreases; when

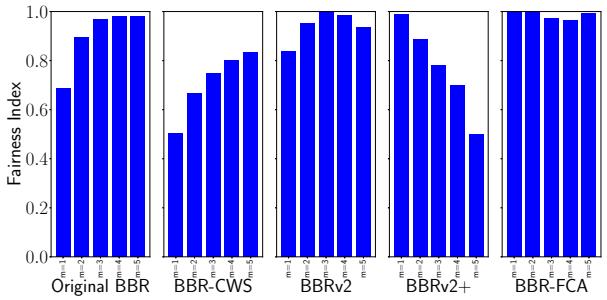

Fig. 12: Fairness index among multiple (m) CUBIC flows and one BBR flow (propagation delay between terminals: 40 [msec], buffer size: 64 [BDP]).

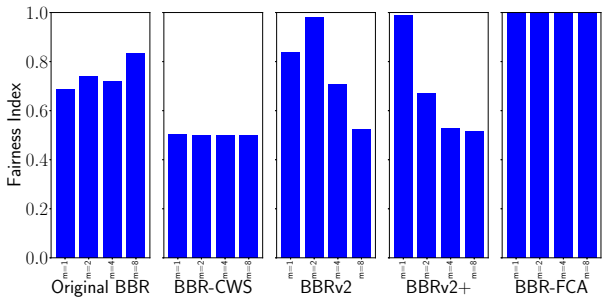

Fig. 13: Fairness index among multiple (m) CUBIC flows and multiple (m) BBR flows (propagation delay between terminals: 40 [msec], buffer size: 64 [BDP]).

BBR-FCA is used, the fairness index among those flows is improved. Specifically, BBR-FCA improves the fairness index by -1.7% to 45.6%, comparing with the original BBR.

Fig. 13 shows the Jain's Fairness Index of CUBIC and BBR flows when the same number ($= m$) of such flows co-exist in the network where the propagation delay of the bottleneck link is 38 [msec] and the buffer size is set to 64 [BDP]. The original BBR has worse results than the results when the propagation delay of the bottleneck link is 18 [msec] in Fig. 9. BBR-CWS, BBRv2, BBRv2+ and BBR-FCA have the results similar to the results in Fig. 9. With both BBRv2 and BBRv2+, the fairness decreases when the number of flows is large as well as in Fig. 9. On the other hand, with BBR-FCA, the fairness is close to 1.0 regardless the number of flows.

According to our other simulations, we confirmed that the magnitude of the bandwidth of the bottleneck link also has little effect on the throughput fairness among CUBIC flows and BBR-FCA flows.

D. Validation for Inheritance of BBR's Characteristics

Finally, we evaluate the average RTT of flow 1, which is the flow transmitted from the first sender, and total throughput when only m ($\geq$ 1) BBR (the original BBR, BBR-CWS, BBRv2, BBRv2+, or BBR-FCA) flows share the bottleneck link in the network where the propagation delay of the bottleneck link is 18 [msec] and the buffer size is set to 64 [BDP]. Fig. 14 shows the results of the average RTT and total throughput. BBRv2+ does not inherit the good

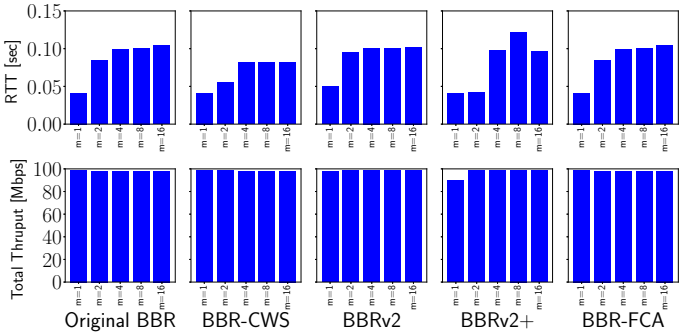

Fig. 14: Average RTT and total throughput of multiple (m) BBR flows (propagation delay between terminals: 20 [msec], buffer size: 64 [BDP]).

performance characteristics of RTT and throughput of the original BBR, while BBR-CWS and BBRv2 almost inherit the performance characteristics of the original BBR, and BBR-FCA inherits them completely.

VII. Conclusion

In this paper, we proposed a modification to the BBR congestion control algorithm, named BBR-FCA, to improve throughput fairness with competing CUBIC flows. We also constructed a novel analytical model of the packet sending rate of a competing CUBIC flow. We evaluate the accuracy of our analytical model and performance of BBR-FCA through simulations. Our results indicate that BBR-FCA flows can improve throughput fairness with competing CUBIC flows regardless of the network parameters and the number of competing flows, except when the buffer size of the bottleneck link is quite large and retransmission timeouts occur frequent.

One of our future works will be to take account retransmission timeouts into our analytical model. The second one is exploring methods of maintaining throughput fairness with congestion control algorithms other than CUBIC.

Acknowledgement

This work was supported by JSPS KAKENHI Grant Number JP20K11786 and JP21KK0202, and the Cooperative Research Project Program (H31/A24) of the Research Institute of Electrical Communication, Tohoku University.

References

[1] I. Kunze, J. Rüth, and O. Hohlfeld, "Congestion control in the wild-investigating content provider fairness," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 1224–1238, 2019.

[2] S. Ha, I. Rhee, and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," *ACM SIGOPS operating systems review*, vol. 42, no. 5, pp. 64–74, 2008.

[3] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-based congestion control," *ACM Queue*, vol. 14, no. 5, pp. 20–53, 2016.

[4] A. Mishra, X. Sun, A. Jain, S. Pande, R. Joshi, and B. Leong, "The great internet TCP congestion control census," in *Proc. of the 2020 SIGMETRICS/Performance Joint International Conference on Measurement and Modeling of Computer Systems*, 2020, pp. 59–60.

[5] R. Ware, M. K. Mukerjee, S. Seshan, and J. Sherry, "Modeling BBR's interactions with loss-based congestion control," in *Proc. of Internet Measurement Conference*, pp. 137–143, 2019.

[6] G. Hasegawa, K. Kurata, and M. Murata, "Analysis and improvement of fairness between TCP Reno and Vegas for deployment of TCP Vegas to the Internet," in *Proc. of IEEE International Conference on Network Protocols*, pp. 177–186, 2000.

[7] M. Hock, R. Bless, and M. Zitterbart, "Experimental evaluation of BBR congestion control," in *Proc. of IEEE International Conference on Network Protocols*, pp. 1–10, 2017.

[8] B. Jaeger, D. Scholz, D. Raumer, F. Geyer, and G. Carle, "Reproducible measurements of TCP BBR congestion control," *Computer Communications*, vol. 144, pp. 31–43, 2019.

[9] P. Hurtig, H. Haile, K.-J. Grinnemo, A. Brunstrom, E. Atxutegi, F. Liberal, and Å. Arvidsson, "Impact of TCP BBR on CUBIC traffic: A mixed workload evaluation," in *Proc. of IEEE International Teletraffic Congress*, vol. 1, pp. 218–226, 2018.

[10] N. Cardwell, Y. Cheng, S. H. Yeganeh, I. Swett, V. Vasiliev, P. Jha, Y. Seung, M. Mathis, and V. Jacobson, "BBRv2: A model-based congestion control," in *Presentation in ICCRG at IETF 104th meeting*, 2019.

[11] G.-H. Kim, Y.-J. Song, and Y.-Z. Cho, "Improvement of inter-protocol fairness for BBR congestion control using machine learning," in *Proc. of 2020 International Conference on Artificial Intelligence in Information and Communication*, pp. 501–504, 2020.

[12] Y.-J. Song, G.-H. Kim, and Y.-Z. Cho, "Improvement of cyclic performance variation between TCP BBR and CUBIC," in *Proc. of 2019 25th Asia-Pacific Conference on Communications*, pp. 1–6, 2019.

[13] ——, "Congestion window scaling method for inter-protocol fairness of BBR," in *Proc. of 2020 IEEE 17th Annual Consumer Communications & Networking Conference*, pp. 1–4, 2020.

[14] F. Yang, Q. Wu, Z. Li, Y. Liu, G. Pau, and G. Xie, "BBRv2+: Towards balancing aggressiveness and fairness with delay-based bandwidth probing," *Computer Networks*, pp. ——, 2022.

[15] Y. Zhang, L. Cui, and F. P. Tso, "Modest BBR: Enabling better fairness for BBR congestion control," in *Proc. of 2018 IEEE symposium on computers and communications*, 2018, pp. 00 646–00 651.

[16] Y.-J. Song, G.-H. Kim, and Y.-Z. Cho, "Enhanced loss-recovery mechanism for BBR to improve inter-protocol fairness," in *Proc. of 2019 International Conference on Information and Communication Technology Convergence*, 2019, pp. 1181–1183.

[17] ——, "Improvement of cyclic performance variation between TCP BBR and CUBIC," in *Proc. of 2019 25th Asia-Pacific Conference on Communications*, 2019, pp. 1–6.

[18] J. Gettys and K. Nichols, "Bufferbloat: Dark buffers in the Internet," *ACM Queue*, vol. 9, no. 11, pp. 40–54, 2011.

[19] W. Bao, V. W. Wong, and V. C. Leung, "A model for steady state throughput of TCP CUBIC," in *Proc. of IEEE Global Telecommunications Conference*, pp. 1–6, 2010.

[20] S. Poojary and V. Sharma, "Analytical model for congestion control and throughput with TCP CUBIC connections," in *Proc. of IEEE Global Telecommunications Conference*, pp. 1–6, 2011.

[21] R. I. L. Goyzueta and Y. Chen, "A deterministic loss model based analysis of CUBIC," in *Proc. of IEEE International Conference on Computing, Networking and Communications*, pp. 944–949, 2013.

[22] G. Vardoyan, N. S. Rao, and D. Towsley, "Models of TCP in high-BDP environments and their experimental validation," in *Proc. of IEEE International Conference on Network Protocols*, pp. 1–10, 2016.

[23] G. Vardoyan, C. Hollot, and D. Towsley, "Towards stability analysis of data transport mechanisms: A fluid model and an application," in *Proc. of IEEE Conference on Computer Communications*, pp. 666–674, 2018.

[24] S. Poojary and V. Sharma, "An asymptotic approximation for TCP CUBIC," in *Proc. of Queueing Systems*, vol. 91, no. 1-2, pp. 171–203, 2019.

[25] L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control (BIC) for fast long-distance networks," in *Proc. of IEEE International Conference on Computer Communications*, 2004, pp. 2514–2524.

[26] A. A. Philip, R. Ware, R. Athapathu, J. Sherry, and V. Sekar, "Revisiting TCP congestion control throughput models & fairness properties at scale," in *Proc. of the 21st ACM Internet Measurement Conference*, 2021, pp. 96–103.

[27] M. Allman, V. Paxson, and E. Blanton, TCP congestion control, RFC 5681, 2009.

[28] The Network Simulator - ns-2, https://www.isi.edu/nsnam/ns/, 2011.

[29] R. Jain, D. Arjan, and B. Gojko, "Throughput fairness index: An explanation," *ATM Forum contribution*, vol. 99, no. 45, pp. ——, 1999.