Improving per-flow fairness by ML-based estimation of competing flows' congestion control algorithm

Keito Maeta
Graduate School of Information Sciences
Tohoku University
Sendai, Japan
maeta.keito.t5@dc.tohoku.ac.jp

Gen Kitagata and Go Hasegawa
Research Institute of Electrical Communication
Tohoku University
Sendai, Japan
{minatsu, hasegawa}@riec.tohoku.ac.jp

***Abstract*—A variety of Internet congestion control algorithms have been developed to overcome the ever-increasing diversity of the Internet. As a result, more than ten different congestion control algorithms co-exist on the current Internet. Therefore, flows with different congestion control algorithms must compete with the bottleneck link, causing unfair bandwidth share. In this paper, we aim to improve per-flow fairness in such situations. In the proposed method, a flow estimates congestion control algorithms of competing flows by a machine learning-based estimation and majority vote algorithm. It then changes its congestion control algorithm based on the estimation results. We evaluated the performance of the proposed method by extensive experiments and found that the estimation accuracy of the proposed method was significantly larger than the chance level and that the per-flow fairness was improved by at most 77.9 [%].**

***Index Terms*—Congestion Control, Transmission Control Protocol (TCP), Fairness, Machine Learning**

I. Introduction

The Internet is becoming increasingly diverse due to the spread of mobile communication terminals such as smartphones and tablet PCs and the growing scale and speed of the network. Congestion control is the source of stability and robustness of the Internet. Therefore, various congestion control algorithms have been developed in response to changes in the Internet. Many of them are rule-based methods designed by the researchers and based on heuristic approaches. Some of these congestion control algorithms are designed for global Internet environments [1]–[3], while others are designed for specific environments [4]–[6].

CUBIC [1] and BBR [2] are the leading congestion control algorithms on the current Internet. In [7], the authors reported that CUBIC has the highest share of 36 [%], followed by BBR with a share of 22 [%]. In other words, the remaining 42 [%] flows employ different congestion control algorithms. In addition, 17 different congestion control algorithms are implemented in the kernel of the Linux OS. Also, congestion control algorithms based on machine learning have been studied extensively in recent years [8]–[11]. Most of them build the learned model for optimal congestion window control from observed parameters on the network environment. Existing machine learning-based methods such as Remy [9] and TCP-Drinc [10] have achieved higher throughput and lower latency than rule-based methods in some network environments.

In light of the above-mentioned background, we believe that it will be difficult to consolidate congestion control algorithms used on the Internet into one in the future. Therefore, situations are inevitable where multiple flows with different congestion control algorithms compete with each other at network bottlenecks. The performance of such flows in terms of throughput and round-trip time (RTT) depends on the combination of algorithms and network environmental parameters, sometimes causing large unfairness. In the past literature [12]–[14], methods to improve the fairness among different congestion control algorithms have been considered. However, most of them have been limited to specific combinations of algorithms and in particular environments. Essentially, it is difficult to maintain fairness among different congestion control algorithms.

The objective of this paper is to improve per-flow fairness in such situations. In the proposed method, a flow estimates congestion control algorithms of competing flows. It then changes its congestion control algorithm based on the estimation results. The proposed method exploits a machine-learning algorithm to determine congestion control algorithms of competing flows based on observations of the network and flow states. We employ an online estimation method that performs the estimation continuously in an active flow. To assess the performance of the proposed method, we conduct extensive experiments with the emulated network environment. We evaluate the proposed estimation method's accuracy and per-flow fairness when changing the congestion control algorithm based on the estimation results.

The remainder of this paper is organized as follows. We first summarize the related work and motivations of this work in Section 2. Section 3 describes an online estimation method of congestion control algorithms of competing flows. In Section 4, fairness improvement is evaluated by changing the congestion control algorithm based on the estimation results. Finally, Section 5 summarizes this study and discusses future issues.

II. Related work and motivations

A. Fairness among congestion control algorithms

Most of the existing congestion control algorithms can be classified into loss-based and delay-based methods. Loss-based methods continuously increase the data sending rate while no packet loss occurs and decrease it when packet losses are

detected. Therefore, when network congestion occurs, packets accumulate in the output buffer of the bottleneck link, causing the increased queuing delay. On the other hand, delay-based methods observe RTTs of sending packets to regulate the data transmission rate. When the RTT increases, the sender decreases the data transmission rate and vice versa. Typical loss-based methods include NewReno [3] and CUBIC. Vegas [15], BBR, and Copa [16] are delay-based methods. These methods perform well when used exclusively, meaning that all flows competing with the bottleneck link use the same algorithm. However, when flows with loss-based and delay-based algorithms co-exist in the bottleneck link, the significant unfairness can be observed [12], [13]. Furthermore, even when flows use the same type of algorithm (loss-based or delay-based), we cannot avoid an unfair share of the bottleneck link bandwidth when they employ different algorithms [17], [18].

Figure 1 shows the experimental results on fairness between two flows competing for the bottleneck link. In the experiment, a flow uses BBR, and another flow chooses one algorithm from BBR, Vegas, CUBIC, and NewReno. The bottleneck link speed is 10 [Mbps], and the output buffer size is 100 [packets] or 300 [packets]. The RTT of both flows without queueing delay is 20 [msec]. The graph in Fig. 1 plots the goodput of the two flows. As shown in this figure, when a BBR flow and another flow with a different algorithm share the bottleneck link, the goodput share differs depending on the characteristics of the bottleneck link and the combination of co-existing algorithms.

Much research has been done to improve fairness among flows using various congestion control algorithms. In [12], Modest BBR is proposed to maintain fairness against competing CUBIC flows by configuring the data sending rate lower than the observed available bandwidth. The authors in [13] introduce BBR-CWS, which improves the fairness between BBR and CUBIC flows, especially when the buffer size of the bottleneck link is small. ArtaVegas, proposed in [14], improves the fairness of Vegas flows competing with Reno flows by modifying the congestion window behavior. These studies focus on the fairness between two specific algorithms, and it is difficult to be deployed on the Internet, where a variety of congestion control algorithms co-exist. In addition, congestion control methods based on machine learning-based algorithms [9]–[11], [19] aim to optimize the performance of their flows, and fairness with competing flows is not considered.

B. Estimation of congestion control algorithms

As a study on the estimation of flow congestion control algorithms, Vegas+ proposed in [20] normally behaves as Vegas and changes to Reno when a flow detects competing Reno flows. Compound TCP (CTCP) proposed in [21] is a hybrid method that combines loss-based and delay-based methods. A CTCP flow switches its algorithm from delay-based to loss-based when it detects competing loss-based flows by observing RTTs. The method in [22] exploits supervised machine learning to estimate whether the competing flow's algorithm is BBR or CUBIC. Most of these studies focus on

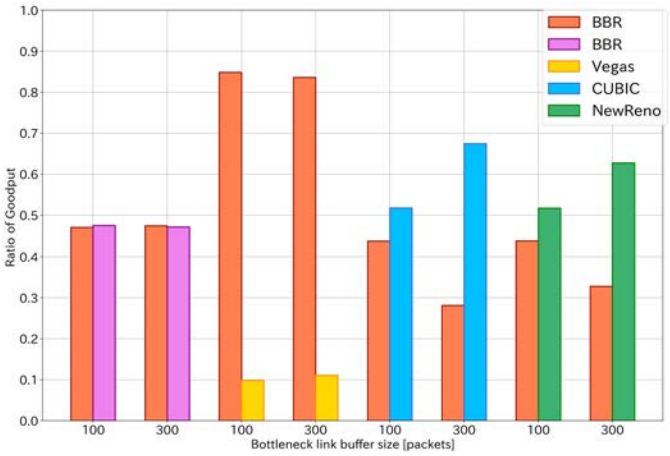

Fig. 1. Fairness among two flows with identical or different congestion control algorithms

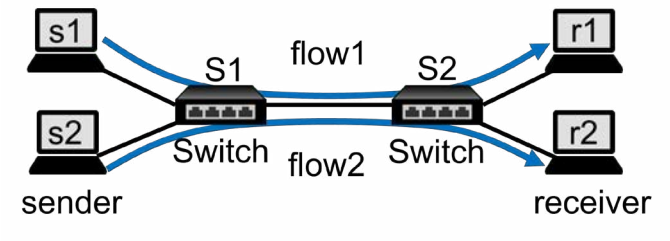

Fig. 2. Network topology

fairness between two specific algorithms and do not consider fairness with other congestion control algorithms.

In [23], a method for estimating a congestion control algorithm of a flow by observing packets passing through intermediate nodes is proposed. However, since the observation is performed at intermediate nodes, it is difficult to be deployed as compared with sender-based methods.

III. Estimating competing flows' congestion control algorithm

A. Overview

We consider a situation where two flows share the bottleneck link, as shown in Fig. 2. Flow 1 from s1 to r1 estimates the congestion control algorithm of flow 2 from s2 to r2. The proposed method is online: flow 1 continuously observes the state of the flow itself and the network condition and conducts the machine learning-based estimation. We then determine the congestion control algorithm of flow 2 by the majority vote algorithm from the recent estimation results.

B. Machine learning-based estimation

We exploit the Random Forest algorithm [24] for the estimation due to its simpleness and fast calculation speed. Table I summarizes the features of the proposed estimation method. We take an exponential weighted moving average for each feature. Note that all features can be obtained from

TABLE I
FEATURES FOR RANDOM FOREST ESTIMATION

The ratio of the largest sRTT (smoothed RTT) to the smallest RTT during the last kRTT (k times the duration of RTT)
The ratio of the largest goodput to the smallest goodput during the last kRTT
The ratio of the latest observed RTT to sRTT
The ratio of the latest observed goodput to the average goodput.
The number of packets sent during the last kRTT
The number of retransmitted packets during the last kRTT
The transmission interval of packets
The reception interval of acknowledgment packets

TABLE II
NETWORK PARAMETER SETTINGS

Access link	
Bandwidth [Mbps]	20
One-way propagation delay time [ms]	1
Buffer Size [packets]	20000
Loss rate [%]	0
Bottleneck link	
Bandwidth [Mbps]	1-10
One-way propagation delay time [ms]	1-100
Buffer Size [Bandwidth-Delay Product]	10^0-10^2
Loss rate [%]	0

the observation by a sender host. The features on RTT and goodput are chosen because they are directly related to the fairness among flows. Also, such values fluctuate depending on the combination of the congestion control algorithms of competing flows and network conditions. The feature value calculation and the estimation are conducted every time the sender receives a new acknowledgment packet.

C. Majority vote algorithm

Since the estimation results with the method mentioned above fluctuate by estimation errors, and we continuously obtain estimation results during the flow, we introduce the majority vote algorithm to determine the competing flow's congestion control algorithm. Figure 3 depicts the algorithm, where B, V, C, and N represent BBR, Vegas, CUBIC, and NewReno, respectively. We select the most frequently estimated algorithm from the recent estimation results with window size w.

D. Evaluation settings

We conducted extensive experiments and evaluated the accuracy of the estimation method. The experiments are conducted with the network emulator Mininet [25]. The network topology is depicted in Fig. 2, and the parameters are summarized in Tab. II. In each experiment, we randomly set the bandwidth, one-way propagation delay, and output buffer size of the bottleneck link between two switches, S1 and S2. Then two flows transmit data packets for 60 [sec]. Flows 1 and 2 use the congestion control algorithm selected from BBR, Vegas, CUBIC, and NewReno. The features are calculated from the packet capture at the network interface of the sender s1.

We executed 2,000 experiments and collected 100 sets of features from each experiment. We extracted training and validation data from 1,000 experiments and test data from the remaining 1,000 experiments. The training and validation data were generated by randomly selecting 80 [%] and 20 [%] of feature sets from 1,000 experiments, respectively. In the model validation, we used the grid search to choose the hyperparameters of the number of trees and the tree depth from 10–600 and 1–30, respectively. Note that the estimation model is constructed for each congestion control algorithm of flow 1.

E. Evaluation results and discussions

Table III summarizes the average estimation accuracy of the Random Forest algorithm described in Subsection III-B. We can see from these results that the estimation accuracy is much higher than the chance level (=25 [%]) regardless of the combination of the congestion control algorithms of flows 1 and 2. It means that the machine learning-based algorithm has enough performance. Especially, when flow 1 uses Vegas, the estimation accuracy is better than other three algorithms. This is because the performance of Vegas can be easily affected by the behavior of competing flows due to its conservative congestion window control. Figure 4 plots the distribution of the estimation accuracy when the congestion control algorithm of flow 1 is BBR. We can see from this figure that the accuracy distribution is quite different when the congestion control algorithm of flow 2 changes. This is because the network environmental parameters, such as the bandwidth, propagation delay, and buffer size of the bottleneck link, affect the estimation accuracy.

Table IV summarizes the accuracy of determining the congestion control algorithm of flow 2 by the majority vote algorithm explained in Subsection III-C when the congestion control algorithm of flow 1 is BBR. Note that w=1 means that the majority vote algorithm is not used. From these results, we can observe that the estimation accuracy improves by introducing the majority vote. Figure 5 plots the distribution of the estimation accuracy with the majority vote algorithm. We can see from this figure that when the accuracy of the Random Forest algorithm is higher, the majority vote algorithm much improves the estimation accuracy. However, when the accuracy of the Random Forest algorithm is lower, the majority vote algorithm degrades the estimation accuracy. The threshold of the two behaviors is around 30 [%] of the accuracy. Consequently, we should avoid such lower accuracy of machine learning-based algorithms to take advantage of the majority vote algorithm.

IV. EVALUATION OF PER-FLOW FAIRNESS

In this section, we present the evaluation results of per-flow fairness when a flow changes its congestion control algorithm to be identical to the estimated algorithm of the competing flow.

A. Evaluation methodology

The evaluation environment for the evaluation is identical to that in Section 3, where two flows share the bottleneck link

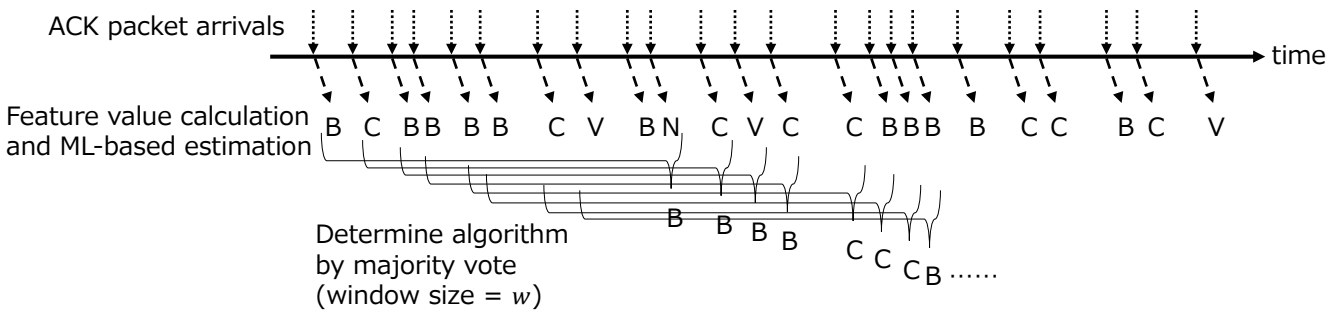

Fig. 3. Majority vote algorithm from recent estimation results

TABLE III
AVERAGE ESTIMATION ACCURACY

		Accuracy [%]			
		flow2			
		BBR	Vegas	CUBIC	NewReno
flow1	BBR	58.1	55.4	46.2	40.3
	Vegas	65.8	68.5	55.3	54.2
	CUBIC	53.4	50.0	45.6	58.1
	NewReno	56.1	48.8	43.8	49.5

TABLE IV
AVERAGE ACCURACY OF THE MAJORITY VOTE ALGORITHM

	Accuracy [%]
$w = 1$	50.0
$w = 10$	53.3
$w = 20$	55.2
$w = 30$	56.3
$w = 40$	57.0
$w = 50$	57.6

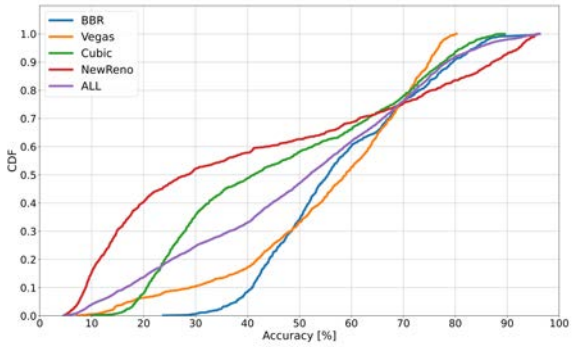

Fig. 4. Distribution of estimation accuracy of the Random Forest algorithm

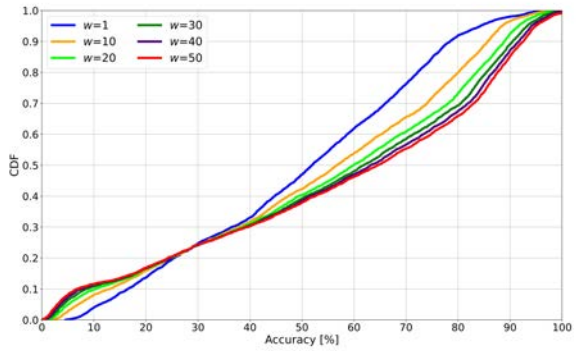

Fig. 5. Distribution of estimation accuracy of the majority vote algorithm

between two nodes, as depicted in Fig. 2. Flow 1 continuously estimates the congestion control algorithm according to the algorithm described in Subsections III-A, III-B, and III-C. We then calculate the goodput of flows 1 and 2 when flow 1 changes its congestion control algorithm to identical to the estimated algorithm of flow 2. In detail, the goodputs of flows 1 and 2 when the congestion control algorithm of flow 2, indicated by j, is either BBR, Vegas, CUBIC, or NewReno are denoted as G_j^1 and G_j^2, respectively. They are calculated according to the following equations:

$$G_j^1 = \sum_{i=\text{BBR, Vegas, CUBIC, NewReno}} \frac{T_i}{T} g_1(i,j) \quad (1)$$

$$G_j^2 = \sum_{i=\text{BBR, Vegas, CUBIC, NewReno}} \frac{T_i}{T} g_2(i,j) \quad (2)$$

where T is the duration of the experiment and T_i is the duration in which the congestion control algorithm of flow 1 is i. $g_1(i,j)$ and $g_2(i,j)$ are the goodput of flows 1 and 2 obtained by the experiments where the congestion control algorithms of flows 1 and 2 are i and j, respectively, where i and j are BBR, Vegas, CUBIC, or NewReno. For these calculations, we conducted experiments for all combinations of two congestion control algorithms of flows 1 and 2, and obtained $g_1(i,j)$ and $g_2(i,j)$ for all combinations of i and j. We assume that the congestion control algorithm of flow 1 is randomly selected when the data transmission starts.

We then evaluate the fairness between two flows using the following two metrics. One is *goodput difference* as defined in

the following equation:

$$D = \frac{G_j^1 - G_j^2}{G_j^1 + G_j^2} \tag{3}$$

This metric ranges from -1 to 1, and when it equals zero, the fairness between two flows is perfect: the goodput of two flows is identical. Otherwise, when it is -1 or 1, flow 2 or flow 1 completely occupies the bottleneck link bandwidth, respectively. Another metric is *absolute goodput difference*, defined as $|D|$. When it is zero, the fairness between two flows is perfect. Otherwise, the larger value means the degraded fairness, regardless of either flow 1 or 2 occupies the bottleneck link.

In the following results, we set w to 10, or 50. For comparative purposes, we show the evaluation results of w=1, meaning that the majority vote is not used.

B. Evaluation results and discussions

Figure 6 plots the distribution of goodput difference defined in Eq. (3) when the congestion control algorithm of flow 2 is BBR, Vegas, CUBIC, and NewReno. The distributions are presented as box plots, where 0, 25, 50, 75, and 100 [%]-values are plotted. The graph includes the results with three values for w, and those when the proposed method is not used, meaning that flow 1 does not change the congestion control algorithm during the experiment.

Overall, when flow 2 uses Vegas, flow 1 outperforms flow 2. On the other hand, when flow 2 uses CUBIC or NewReno, flow 2 obtains better goodput than flow 1. This is consistent with previous works on the fairness between loss-based and delay-based algorithms [20], [21].

Comparing the results without the proposed method ("Not used" in the graph) and those of the proposed method without the majority vote algorithm ("w=1"), we can observe that the introduction of the machine learning-based estimation significantly improves the fairness between the two flows especially when flow 2 uses Vegas. However, when flow 2 uses CUBIC or NewReno, the 50 [%] value of goodput difference degrades, while its variance becomes small. When the majority vote algorithm is applied ("w=10" and "w=50"), the throughput difference approaches zero in all cases of the congestion control algorithm of flow 2. These results exhibit the effectiveness of the combination of machine learning-based estimation and majority vote algorithms.

Figure 7 plots the average values of absolute goodput difference. From this figure, we can see that the proposed method significantly improves the fairness between two flows, regardless the majority vote algorithm is used or not. In addition, when introducing the majority vote algorithm, the fairness is further enhanced, especially when the window size for the majority vote, w, is set to a larger value. In detail, the absolute throughput difference is enhanced by 77.9 [%] when flow 2 uses BBR and w=50. However, in general, the large value of w would degrade the transient behavior of the proposed method, especially when the congestion control algorithm of competing flow changes. The appropriate setting of w is one of our future works.

V. Conclusion and Future Work

This paper proposed a method to improve per-flow throughput fairness by introducing machine learning-based estimation of competing flow's congestion control algorithm. The proposed method exploits the Random Forest algorithm for the estimation from observation of the network conditions and flow states. It then determines the congestion control algorithm of the competing flow by the majority vote algorithm. Extensive experiments and numerical evaluations revealed that the estimation accuracy is much higher than the chance level, and the fairness can be improved by up to 77.9 [%] when the congestion control algorithm is changed to the estimated one.

In future work, the evaluation of the proposed method in more realistic situations is essential. Specifically, we will estimate the number of flows when the competing flows increase. Furthermore, the evaluation will be performed on complex networks. Also, we plan to construct a novel congestion control algorithm based on the proposed method in this paper.

References

[1] S. Ha, I. Rhee, and L. Xu, "CUBIC: a new TCP-friendly high-speed TCP variant," *ACM SIGOPS OSR*, vol. 52, pp. 64–74, Jul. 2008.

[2] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-based congestion control," *ACM QUEUE*, vol. 14, pp. 20–53, Dec. 2016.

[3] T. Henderson, S. Floyd, A. Gurtov, and Y. Nishida, "The NewReno modification to TCP's fast recovery algorithm," *Request for comments 6582*, Apr. 2012.

[4] K. W. A. Sivaraman and H. Balakrishnan, "Stochastic forecasts achieve high throughput and low delay over cellular networks," in *Proceedings of the 10th USENIX Symposium on NSDI*. USENIX, Apr. 2013, pp. 459–471.

[5] C. P. Fu and S. C. Liew, "TCP Veno: TCP enhancement for transmission over wireless access networks," *IEEE Journal on selected areas in communications*, vol. 21, no. 2, pp. 216–228, Feb. 2003.

[6] R. Mittal, V. Lam, N. Dukkipati, E. Blem, H. Wassel, M. Ghobadi, A. Vahdat, Y. Wang, D. Wetherall, and D. Zats, "TIMELY: RTT-based congestion control for the datacenter," in *Proceedings of ACM SIGCOMM 2015*, Aug. 2015.

[7] M. Ayush, S. Xiangpeng, J. Atishya, P. Sameer, J. Raj, and L. Ben, "The great Internet TCP congestion control census," in *Proceedings of the ACM on MACS*, vol. 3. ACM New York NY USA, Dec. 2019, pp. 1–24.

[8] T. Zhang and S. Mao, "Machine learning for end-to-end congestion control," *IEEE Communications Magazine*, vol. 58, no. 6, pp. 52–57, Jul. 2020.

[9] K. Winstein and H. Balakrishnan, "TCP ex machina: Computer-generated congestion control," in *Proceedings of ACM SIGCOMM 2013*, Aug. 2013, pp. 123–134.

[10] K. Xiao, S. Mao, and J. K. Tugnait, "TCP-Drinc: Smart congestion control based on deep reinforcement learning," *IEEE Access*, vol. 7, pp. 11 892–11 904, Jan. 2019.

[11] N. Jay, N. Rotman, B. Godfrey, M. Schapira, and A. Tamar, "A Deep Reinforcement Learning Perspective on Internet Congestion Control," in *Proceedings of ICML 2019*, Jun. 2019.

[12] Y. Zhang, L. Cui, and F. P. Tso, "Modest BBR: Enabling better fairness for BBR congestion control," in *Proceedings of IEEE ISCC 2018*, Nov. 2018, pp. 646–651.

[13] Y.-J. Song, G.-H. Kim, and Y.-Z. Cho, "Congestion window scaling method for inter-protocol fairness of BBR," in *Proceedings of IEEE CCNC 2020*, Jan. 2020, pp. 1–4.

[14] N. Afraz and M. Analoui, "TCP-ArtaVegas: Improving the fairness of TCP-Vegas," in *Proceedings of 2015 23rd ICEE*, May 2015, pp. 761–764.

[15] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End to end congestion avoidance on a global Internet," *IEEE J-SAC*, vol. 13, no. 8, pp. 1465–1480, Oct. 1995.

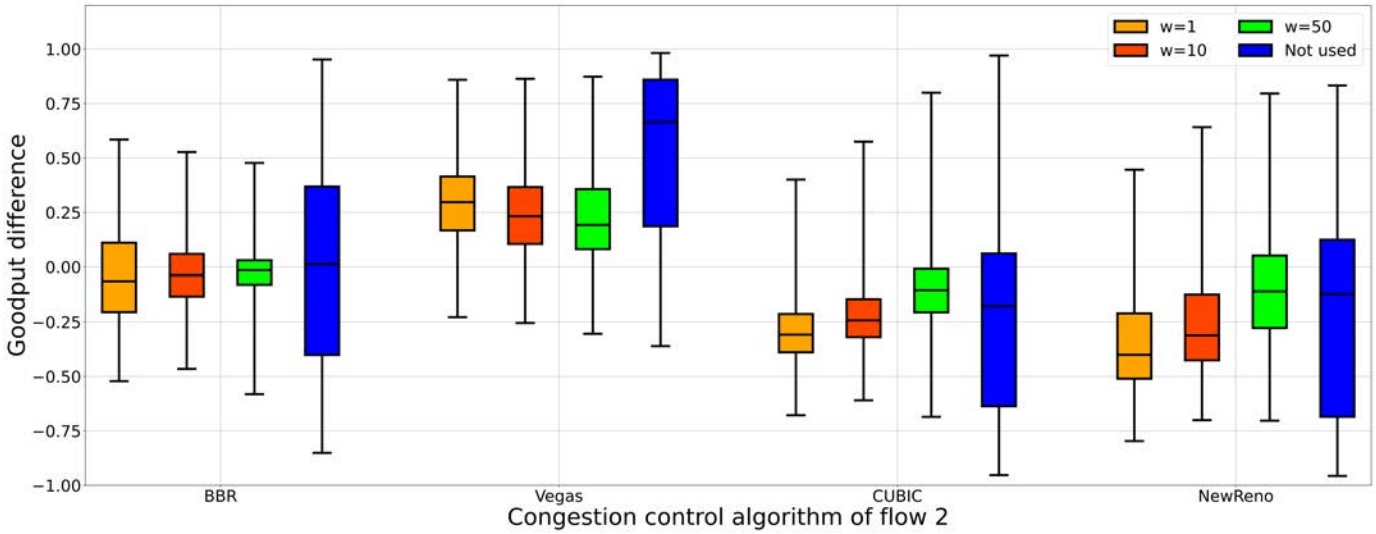

Fig. 6. Goodput difference

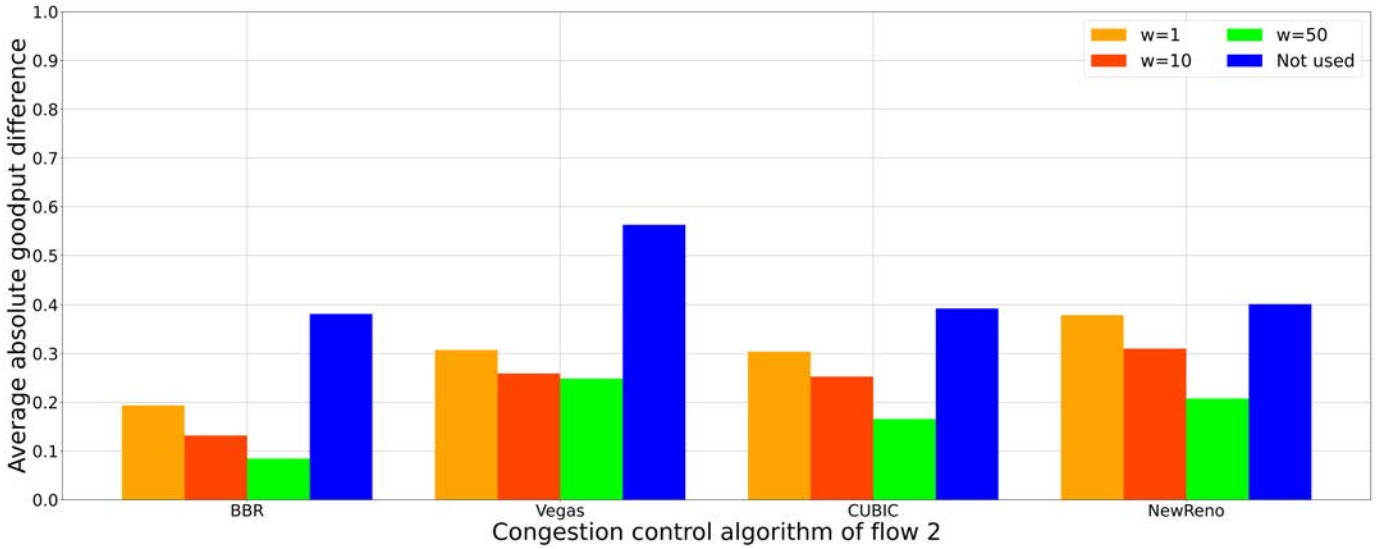

Fig. 7. Absolute goodput difference

[16] V. Arun and H. Balakrishnan, "Copa: Practical Delay-Based Congestion Control for the Internet," in *Proceedings of USENIX NSDI 2018*, Apr. 2018.

[17] Z. Chi-Min, "Fairness Improvement of High Speed TCP Congestion Control Algorithm," in *Proceedings of 2013 ICCIS*. IEEE, Jun. 2013, pp. 1130–1133.

[18] I. Abdeljaouad, H. Rachidi, S. Fernandes, and A. Karmouch, "Performance analysis of modern TCP variants: A comparison of Cubic, Compound and New Reno," in *Proceedings of 2010 25th QBSC*. IEEE, May 2010, pp. 80–83.

[19] N. Dukkipati, Y. Cheng, and A. M. Vahdat, "QTCP: Adaptive congestion control with reinforcement learning," *IEEE TNSE*, vol. 6, July-Sept 2019.

[20] 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 *Proceedings IEEE ICNP 2000*, Nov. 2000, pp. 177–186.

[21] T. Kun, S. Jingmin, Z. Qian, and S. Murad, "A compound TCP approach for high-speed and long distance networks," in *Proceedings of IEEE INFOCOM 2006*, Feb. 2006, pp. 1–12.

[22] G.-H. Kim, Y.-J. Song, and Y.-Z. Cho, "Improvement of inter-protocol fairness for BBR congestion control using machine learning," in *Proceeding of IEEE ICAIIC 2020*, Feb. 2020, pp. 501–504.

[23] D. H. Hagos, P. E. Engelstad, A. Yazidi, and Ø. Kure, "General TCP state inference model from passive measurements using machine learning techniques," *IEEE Access*, vol. 6, pp. 28 372–28 387, Apr. 2018.

[24] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[25] K. Kaur, J. Singh, and N. S. Ghumman, "Mininet as software defined networking testing platform," in *Proceedings of ICCCS*, Aug. 2014, pp. 139–42.