Extensive Evaluation on the Performance and Behaviour of TCP Congestion Control Protocols under Varied Network Scenarios

Jinting Lin[a], Lin Cui[a,*], Yuxiang Zhang[a], Fung Po Tso[b], Quanlong Guan[a]

[a] *Department of Computer Science, Jinan University, Guangzhou, China*
[b] *Department of Computer Science, Loughborough University, LE11 3TU, UK*

Abstract

In recent decades, many TCP Congestion Control (CC) protocols have been proposed to improve the performance and reliability of TCP in various network scenarios. However, CC protocols are usually closely coupled with network conditions such as latency and packet loss. Considering that networks with different properties are common, e.g., wired/wireless LAN and Long Fat Networks (LFNs), investigating both performance and behaviors of CC protocols under varied network scenarios becomes crucial for both network management and development. In this paper, we conduct a comprehensive measurement study on the goodput, RTT, retransmission, friendliness, fairness, convergence time and stability of most widely-used CC protocols over wired LAN/WAN and wireless LAN (both 2.4GHz and 5GHz Wi-Fi). We also conduct comparative studies with respect to transmission cost, congested reverse path and bottleneck queue size in network simulator.

Based on our analysis, we reveal several interesting and original observations. We found that the goodput of BBR is at least 22.5% lower than other CC protocols in wireless LAN due to insufficient pacing rate, even though it can always fully utilize the bottleneck bandwidth with low RTT in wired networks. We also observed that the total on-wire data volume of BBR is higher

*Corresponding author
Email address: tcuilin@jnu.edu.cn (Lin Cui)

than CUBIC (e.g., 2.37% higher when RTT=100ms and loss rate=0.01%). In addition, BBR can fully utilize the bottleneck bandwidth in most queue sizes (≥20packets). Surprisingly, we noticed that as the default CC protocol in most modern operating systems, CUBIC is too aggressive and unfriendly in both LAN and wireless LAN, greatly suppressing the goodput of other competing CC protocols . More specifically for CUBIC in wireless LAN, it generates 129% more retransmissions than other CC protocols. Nevertheless, we have also seen that, in scenario with heavily-congested reverse path, CUBIC can provide full utilization on bottleneck bandwidth. Lastly, we also observed that BBR converges very quickly in all evaluated scenarios, while other CC protocols present varied results, e.g., Westwood+ and Veno converge faster in 5GHz Wi-Fi networks than 2.4GHz networks.

Keywords: TCP Congestion Control Protocol, Performance Evaluation

---

1. Introduction

Transmission Control Protocol (TCP) is a connection-oriented transport-layer protocol, providing stable and reliable data transfer service for upper layers in TCP/IP protocol stacks. Due to its stable and reliable transportation service, TCP is now one of the most widely-used transport-layer protocol [1].

For a certain TCP connection, network bottlenecks where incoming packets exceed outgoing bandwidth are common in most networks [2]. They can cause packet congestion and eventually packet queue built-up in the buffer of bottleneck. If the congestion could not be mitigated in time, packets could be dropped due to bufferbloat, leading to a congestion collapse [3]. Hence, to efficiently prevent network congestion and stabilize data flow, researchers from both academia and industry have proposed many CC protocols that control packet sending rate [2, 4, 5, 6].

However, many CC protocols are only designed to address network congestion under specific networking conditions in terms of bandwidth, latency, and packet loss rate. For example, BBR [2], Illinois [6], BIC [7], and NewReno

[8] are only designed to fully utilize bottleneck bandwidth in Long Fat Networks (LFNs), whilst Westwood+ [9], and Veno[10] are intended to address network congestion in wireless networks with constant interference. However, as today's networks are becoming much more complex and dynamic, it is also increasingly uncertain how existing CC protocols can cope with such increased network complexity and dynamism. Thus, two main questions concerning the performance (e.g., throughput, RTT, and retransmission) and behaviour (e.g., fairness, friendliness, convergence time and stability) of CC protocols are raised: ***Are commonly-used CC protocols (e.g., CUBIC and BBR[1]) still capable of offering good performance under different wired and wireless network scenarios? And what are their behavioral properties under these network scenarios?***

To understand the performance and behavior offered by existing CC protocols under different network scenarios, many measurements studies have been conducted and published in the literature. Nevertheless, most existing measurement works either focus on the performance of CC protocols under specific network scenario [11, 12], or only study their performance metrics without analyzing their behaviors [13, 14]. Motivated by this, ***we have conducted extensive evaluations and comprehensive analysis on both performance and behaviour of multiple widely-used CC protocols under varied network scenarios, including wired LAN/WAN and wireless LAN (both 2.4GHz and 5GHz Wi-Fi networks)***. Comparing with existing works (Table 1), this paper has made the following contributions:

- ***Contribution 1: We perform comprehensive analysis on the behaviors of investigated CC protocols in addition to performance metrics.*** Based on our evaluation results, several important observations are highlighted and carefully analyzed in this paper. For instance, CUBIC achieves the highest utilization in fairness evaluations with ideal

[1] CUBIC is the default CC protocol in the latest version of Linux, OSX and Windows. And BBR is a recently proposed CC protocol that widely deployed in Google services [2].

share on the bottleneck bandwidth in LAN, but it is very unfriendly in wireless LAN; Hybla and Illinois can provide at least 20% improvement on goodput compared to CUBIC in lossy LFN.

- ***Contribution 2: We provide the first yet detailed analysis on the performance and behaviour of BBR in 5GHz Wi-Fi networks***. Results show that the goodput of BBR is at least 20% lower compared with CUBIC in wireless LAN due to insufficient pacing rate.

- ***Contribution 3: We provide a fine-grained comparison study on the transmission cost of BBR and CUBIC.*** And the result shows that BBR would lead to at most 3.2% more on-wire data volume, and 10X more retransmissions per packet compared with CUBIC in certain case.

- ***Contribution 4:We conduct a set of evaluations to reveal the effect on convergence time brought by different network properties (e.g., RTT and loss rate).*** In wired scenarios, increased RTT would lead to larger convergence time when loss rate$\geq$0.01%, while increased loss rate would lead to smaller convergence time. Besides, in wireless scenarios, Westwood+ and Veno present faster convergence in 5GHz Wi-Fi network, while CUBIC presents slower convergence in 5GHz Wi-Fi network. It is worth noting that, BBR is able to converge at the beginning of evaluations in almost all experiments.

- ***Contribution 5: We carry out comparative evaluations in network simulator to reveal the effect of heavily-congested reverse path and bottleneck queue size.*** The evaluation result shows that CUBIC can fully utilizes the bottleneck bandwidth in link with heavily-congested reverse path, and the goodput of other CC protocols which consider the RTT (delay) is reduced to different levels. In evaluation concerning queue size, BBR provides almost full utilization on most queue size ($\geq$20 packets), and the goodput of other CC protocols is degraded to varying degrees when there is a small queue size ($\leq$100 packets).

The remainders of this paper are organized as follows. In Section 2, we briefly present the metrics, scenarios, and major results of several other measurement studies. Subsequently, we also compare our work with these studies carefully, and discuss the motivation of this paper in this section. After that, an overview of CC protocols investigated in this paper is presented in Section 3. Section 4 details our methodology and evaluation setup. Section 5 and Section 6 present the results and analysis in wired and wireless scenarios respectively. Comparative analyses on transmission cost, convergence time, heavily-congested reverse path and different bottleneck queue size are presented in Section 7. Finally, conclusions and future works can be found in Section 8.

2. Related works and motivation

Since CC protocol controls sending rate of packets, its behavior has decisive impact on the performance of TCP. In light of this, measurement study on CC protocols with the aim of improving the performance of CC protocols has been always a research hotspot. Table 1 presents a summary on the main features of several measurement studies and our work.

In wired network scenarios based on Ethernet LAN and WAN, existing studies mainly focus on evaluating the performance metrics of existing CC protocols, only some of them provide analysis on behavior [1, 11, 12, 15, 16, 17, 18]. Alrshah *et al.* [1] conducted a measurement study on multiple CC protocols under scenarios with different buffer size, mainly revealing that CUBIC and YeAH overcome all other investigated CC protocols in terms of throughput under most investigated scenarios (buffer size$\geq$100 packets). Lukaseder *et al.* [11] evaluated performance of 6 widely-used CC protocols on a state-owned research network with 10Gbps bottleneck bandwidth, demonstrating that BIC presents the highest throughput with unfriendliness against NewReno, while CUBIC serves as a better alternative due to better friendliness. Nguyen *et al.* [12] used ns-3 network simulator to evaluate performance of 8 common-used CC protocols in Data Center Network (DCN), arguing that Vegas outperforms all other CC pro-

Table 1: The Main Feature of Related Performance Evaluation Studies[1]

	# of CC Protocols	Evaluated Metrics					Network Scenarios			
		Performance Metrics[2]	***cwnd***	**Fairness**	**Friendliness**	**Convergence, Stability/ Spread**	**Wired LAN/WAN**	**Wi-Fi**	**Others**	**Comparison Between Scenarios**
[1]	11	Thp. & loss rate	✓	✓	✗	✗	1Gbps Simulated LAN	✗	✗	✗
[11]	6	✓	✗	✓	Against Reno	✓	10Gps WAN Testbed	✗	✗	✗
[12]	8	✓	✗	✗	✗	✗	100Mbps Simulated DCN	✗	✗	✗
[15]	1	✓	✗	✓	Against CUBIC	✗	10Gps & 1Gbps LAN Testbeds	✗	✗	✗
[16]	2	✓	✗	✓	Against CUBIC	✗	10Mbps Simulated LAN	✗	✗	✗
[17]	13	Thp. & RTT	✓	✓	Against CUBIC, Vegas and BBR	✗	100Mbps Testbed	✗	✗	✗
[13]	4	✓	✗	✗	Against CUBIC	✗	1Gbps WAN	2.4GHz	3G	✗
[14]	6	Thp. & Latency	✗	✗	✗	✗	✗	2.4GHz	3G & 4G	✗
[18]	13	Thp.	✗	✓	Against Reno	✗	1Gbps Emulator and Testbed	2.4GHz	Satellite	✓
[19]	11	Thp.	✓	✗	✗	Convergence	✗	2.4GHz & 5GHz	✗	✗
Our Work	13	✓	✓	✓	Against CUBIC & NewReno	✓	1Gbps LAN/WAN[3]	2.4GHz & 5GHz	✗	✓

[1] Generally, checkmark means the authors evaluated the corresponding metric(s) (in group "Evaluated Metrics"), or included the corresponding scenario (in group "Scenarios"), while crossmark means not.
[2] Performance metrics include throughput (Thp.), RTT (latency), and loss rate (retransmission).
[3] Servers in this paper are all equipped with 1Gbps NICs. The bottlenecks in wired LAN and WAN scenarios are set to be 100Mbps or 1Mbps.

tocols in DCN. Hock *et al.* [15] conducted an in-depth evaluation for BBR on testbeds with 10Gbps data rate and 1Gbps bottleneck bandwidth, finding that BBR presents its intended behavior and several severe issues, e.g., increased queuing delay, unfairness and packet loss. Scholz *et al.* [16] carried out a set of comparative evaluations with respect to CUBIC and BBR, mainly revealing that BBR can not share bottleneck bandwidth with CUBIC fairly. Turkovic *et al.* [17] evaluated the interactions between loss-based, delay-based and hybrid-based CC protocols, mainly revealing that the bottleneck bandwidth can not always be shared among these CC protocols, especially with flows with different RTTs. Callegari *et al.* [18] offered a comprehensive analysis on the throughput, fairness and friendliness of 13 CC protocols upon wired, wireless and simulated scenarios, demonstrating that the performance offered by different CC protocols strongly depends on network properties (e.g., RTT and loss rate).

However, to the best of our knowledge, only a few measurement studies based on wireless LAN (i.e., Wi-Fi) have been conducted in recent years [13, 14, 19]. Murray *et al.* [13] evaluated the performance of different CC protocols and showed that CUBIC and Hybla outperform Westwood+ and Veno in both wired and wireless scenarios. Ong *et al.* [14] conducted a set of evaluations on the performance of CC protocols based on Wi-Fi and cellular network, showing that alternative CC protocols, e.g., Westwood+ and YeAH, could not provide significant increase in performance compared with CUBIC. Alakoca *et al.* [19] analyzed the performance of several CC protocols in 2.4GHz (IEEE 802.11n) and 5GHz (IEEE 802.11ac) Wi-Fi networks, mainly presenting that BIC and CUBIC achieve higher and more stable throughput compared with other CC protocols. However, we believe the performance and behaviour of newly-proposed BBR under 5GHz Wi-Fi (IEEE 802.11ac) scenario has not been systematically investigated in any existing measurement study.

Except for Wi-Fi network, cellular network is another wireless scenario that being extensively used. Compared to Wi-Fi network, hosts in cellular network has a larger range of movement and the ability to move at higher velocity. Thus, the range and speed of host movement is often considered as two important

factors in evaluating the performance of TCP (MPTCP) congestion control protocols [20, 21, 22, 23, 24]. Due to the complexity of conducting measurement studies over cellular networks, and gradual rolling out of 5G globally, we plan to design and conduct similar studies in the cellular network in the future.

In this paper, we focus on a more comprehensive and systematic measurement study on CC protocols under varied network scenarios, including Ethernet LAN, WAN, wireless LAN (both 2.4GHz and 5GHz Wi-Fi networks). In comparison with existing studies, we have investigated both performance metrics (i.e., goodput, RTT and retransmission) and behavioral properties (i.e., fairness, friendliness, convergence and spread) on real testbed and Mininet. More importantly our studies have revealed that there is no single CC protocol can consistently perform the best in all scenarios. We believe our new findings will give the research community some new insights into designing future CC protocols.

3. TCP congestion control overview

In this paper, we evaluate the following most widely-used CC protocols: BBR [2], CUBIC [4], Hybla [5], Illinois [6], BIC [7], NewReno [8], Westwood+ [9, 25, 26], Veno [10], HighSpeed [27], Scalable [28], Vegas [29], YeAH [30], and H-TCP [31]. Generally, these CC protocols can be classified into four categories according to their feedbacks: loss-based, delay-based, loss-delay-based (hybrid-based) and Bandwidth-Delay-Product-based (BDP-based). In Table 2, we summarize all these CC protocols and highlight their main features.

Loss-based CC protocols, such as NewReno and CUBIC, take packet loss as the signal of network congestion. For these protocols, packet losses caused by both congestion and random error are all interpreted as network congestion, leading to undesired reduction of congestion window. Alternatively, delay-based CC protocols, such as Vegas and Hybla, use the increase in RTT (an effective equivalence to delay) to identify network congestion.

However, using pure loss-based or delay-based CC protocol may not mitigate

Table 2: TCP congestion control protocols overview

Category	**Protocol**	**Main Feature**
Loss-Based	CUBIC	Improve window control mechanism and friendliness based on BIC
	BIC	Improve performance in LFN
	NewReno	Recover multiple losses occurring in single sending window
	Scalable	Better utilize the high-speed WANs
	HighSpeed	Better performance in network with High-BDP
Delay-Based	YeAH	Balance efficiency, fairness, friendliness and robustness
	Hybla	Use normalized RTT to remove the negative effect from abnormal RTT
	H-TCP	Improve friendliness in legacy network
	Westwood+	Estimate available bandwidth by incoming ACKs in wireless network
	Veno	Use prediction to discriminate loss from random error and congestion in wireless network
	Vegas	Introduce proactive congestion control mechanism
Loss-Delay-Based	Illinois	Provide better performance in heavy-congested network
BDP-Based	BBR	Fully utilize bottleneck bandwidth and keep low queue length

network congestion in some cases [2, 9]. Thus, loss-delay-based CC protocols have been proposed to address this issue. For example, Illinois, a loss-delay-based CC protocol, determines the direction in which the congestion window

should be changed by packet losses, and adjusts its pace by measuring RTT.

In addition to above categories, BBR [2], a BDP-based CC protocol, implements a completely different way to enforce congestion control. It takes the amount of acknowledged packets and the lowest RTT experienced recently to make continuous estimations on bottleneck bandwidth, and sets the congestion window according to these estimations. It has been shown that the throughput of BBR is 25X higher than CUBIC at most in lossy LFN [2].

4. Methodology and evaluation setup

In this section, we describe our experimental setup and methodologies used to conduct experiments.

4.1. Data collection

In this paper, iperf3 [32] (version 3.2) is adopted as the main evaluation tool in most evaluations. It generates TCP flows between sender and receiver, and collects goodput, RTT, retransmission and congestion window. In this version of iperf3, goodput, RTT, retransmission and congestion window are all reported as the average value of each sampling interval. Since iperf3 establishes a control connection to exchange control commands (e.g., the start and end of evaluation), and a data connection to transmit data, we only capture the packets from data connection for further analysis. In addition to iperf3, curl [33] (version 7.54.0) is also used to test the goodput of CC protocols for Web traffic[2]. In this version of curl, it calculates the goodput simply by dividing the size of transferred file with the transfer time after completing the whole file transfer operation.

4.2. Friendliness and fairness

Since it is common for multiple flows to share the same bottleneck, measuring the friendliness and fairness of a CC protocol is an important issue in analyzing

[2] For simplicity, file transfer operation is adopted as a representative of common Web traffic.

its behavior. We refer friendliness to the ability for a CC protocol to share the same bottleneck with *other* CC protocols, while fairness considers this ability within *identical* CC protocols [18].

With the average goodput of n competing flows $[x_1, x_2, ..., x_n]$, both friendliness and fairness can be computed through Jain's Index [34] (J for short), which is defined as

$$J(x_1, x_2, ..., x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} {x_i}^2}. \tag{1}$$

For simplicity, we only consider the situation in which only two flows, say Flow-1 and Flow-2, are competing at the same bottleneck. In this case, J varies from 0.50 to 1.00 (1.00 the optimal situation). According to [18], the threshold of *"reasonable"* friendliness and fairness is set to 0.95, indicating that the ratio of goodput achieved by two flows is roughly 4:6. And the threshold of *"optimal"* friendliness and fairness is set to 0.99 (the ratio of goodput achieved by two flows is roughly 4.5:5.5), since it is almost impossible to achieve exactly the same goodput at bottleneck for all competing flows (i.e., J reaches exactly 1.00).

4.3. Convergence time and stability

Additionally, convergence time and stability are also two important aspects in analyzing the behavior of involved CC protocols. In this paper, we extend two simple yet effective metrics to measure them based on [11, 35].

In [11, 35], the authors use the moving average of throughput to find out the convergence time, and average distance to calculate stability. In this paper, we use the moving average of congestion window that helps us to determine the convergence time based on the behaviour of investigated CC protocol, instead of the goodput. Also, we use standard deviation instead of average distance, because standard deviation leads to less stability in data set with excessive extreme values. Detailed method to calculate the convergence time and stability of investigated CC protocols are explained as below.

First, we define the moving average $\mu(t)$ with respect to the congestion

window of a CC protocol at time t as

$$\mu(t) = \frac{\sum_{i=t-W}^{t} c(i)}{W}, \tag{2}$$

where $c(i)$ is the size of congestion window at time i, and W denotes the length of time window. In practice, the sampling interval between $c(i)$ and $c(i-1)$ is set to 0.1s, and W is set to 30 to cover a window of 30 seconds (the duration of each trial is at least 600s). Then, the average of $\mu(t)$, say $\bar{\mu}$, is easy to calculate. In this case, the convergence time t_c is defined as the time that a CC protocol needs to let its congestion window reach the range bordered by 120% (the upper bound) and 80% (the lower bound) of $\bar{\mu}$ since the start of trial.

As mentioned before, we also compute the stability s of congestion window after t_c by standard deviation defined as

$$s = \sqrt{\frac{1}{N}\sum_{i=t_c}^{t_e}(\mu(i) - \bar{\mu})^2}, \tag{3}$$

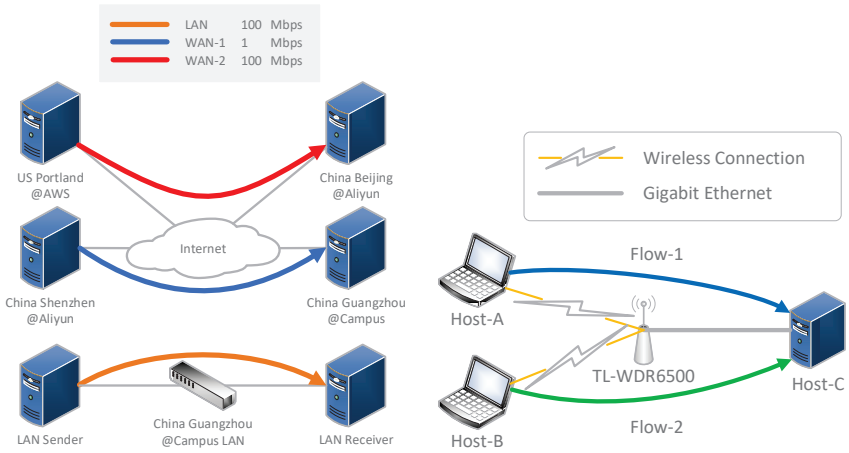
where t_e denotes the time that this trial ends, and N denotes the number of samples between t_c and t_e.

4.4. Test network scenarios

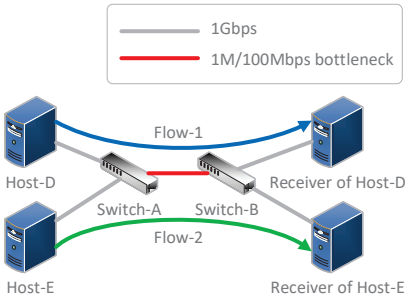
In this section, we describe in details about our topological setups and evaluation scenarios, as follows:

- ***Scenario: LAN.*** This scenario is based on 100/1000Mbps Ethernet LAN within a university campus network (Figure 1(a)). The sender is deployed in the campus data center, and the receiver is in a lab in the same university. Most links along the flow path are 1000Mbps, while a 100Mbps link is introduced as the bottleneck.

- ***Scenario: WAN-1.*** As shown in Figure 1(a), the sender is deployed in Shenzhen China (hosted in Aliyun [36]), and the receiver is deployed in Guangzhou China (hosted in campus data center). The bottleneck is about 1Mbps limited by service provider.



(a) LAN/WAN (b) Wireless LAN



(c) Dumbbell topology in Mininet-based friendliness and fairness evaluation

Figure 1: The network topologies of LAN/WAN, wireless LAN, and simulated dumbbell network. Flow-1 and Flow-2 are the competing flows at the bottleneck.

- ***Scenario: WAN-2.*** In this scenario, the sender is deployed in Portland US (hosted in AWS [37]), while the receiver is located in Beijing China (hosted in Aliyun) as depicted in Figure 1(a). The bottleneck bandwidth is about 100Mbps which is also limited by service providers. Since the link properties of WAN-2 such as bottleneck bandwidth, RTT and loss rate can change dynamically during evaluation, the evaluation results may be distorted. To minimize the effect of distortion, evaluations are repeated multiple times over different days.

- ***Scenario: Wireless LAN.*** To carry out evaluations in wireless LAN, we setup an IEEE 802.11 Wi-Fi network as depicted in Figure 1(b). In this network, four devices are involved: a TP-Link TL-WDR6500 wireless router with 2.4GHz (IEEE 802.11n, max bitrate=300Mbps) and 5GHz (IEEE 802.11ac, max bitrate=866Mbps) enabled, two laptops connected to the router via wireless connection, and one server connected to the router via Gigabit Ethernet. Host-A serves as the sender of Flow-1, and Host-B serves as the sender of Flow-2. Host-C is the receiver of Flow-1 and Flow-2. Since the bandwidth offered by wireless connections are lower than the Gigabit Ethernet, the bottleneck of this scenario is on the wireless connections.

- ***Simulated Dumbbell Topology.*** In Section 5.3, we setup a simulated dumbbell topology through Mininet [38], in order to create a bottleneck for competing flows and exclude the influence from other flows in wired scenarios. Within this topology, four servers and two switches are deployed: two pairs of sender and receiver (Host-D/Host-E and their corresponding receivers), as well as Switch-A/Switch-B. The links between servers and switches are with 1Gbps bandwidth capacity (effectively creates a bottleneck at the inter-switch link with 100Mbps bottleneck bandwidth capacity), no propagation delay and replication error. Link properties of inter-switch link (bottleneck) are configured according to the performance baselines obtained from Section 5.1.1.

In aforementioned scenarios, servers and laptops are all running on 4-core Intel CPU and 4GB DDR3 memory with Ubuntu Desktop 16.04 LTS (Linux kernel version 4.9) operating system. To avoid any explicit or implicit network optimization between different data centers within the same service provider, this paper chooses AWS and Aliyun to host servers in WANs separately. As default system configurations, SACK (Selective ACK) [39], FACK (Forward ACK) [40] and DSACK (Delayed Selective ACK) [41] are all activated, while ECN (Explicit Congestion Notification) [42] is set as receive-only. Likewise, the default

settings of internal parameters (e.g., the pacing rate of BBR) for all investigated CC protocols are also unchanged in this paper. Furthermore, the size of TCP buffers is controlled by Linux system automatically in all servers and laptops (the minimum size is 4KB, and the maximum size is 6MB). In wireless scenarios, RTS/CTS is deactivated by setting the RTS threshold to 2347 according to the default configuration of most modern mobile devices and access points [43, 44, 45]. Finally, in both wired/wireless scenarios, the size of TX/RX queues on NICs use default value, i.e., 1000 packets.

5. Evaluation in LAN/WAN

In this section, we present the evaluation results and analysis on the performance and behaviour of CC protocols in wired scenarios: LAN, WAN-1 and WAN-2. Since Westwood+ and Veno are designed for wireless networks, they are analyzed in Section 6.

5.1. TCP Performance Evaluation

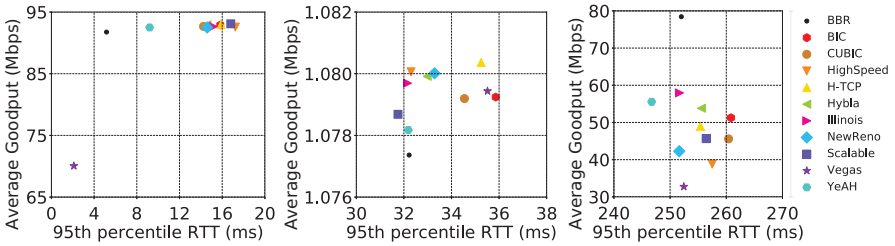
We first focus on evaluating the goodput, RTT and retransmission of different CC protocols based on LAN/WAN. Unless otherwise specified, each CC protocol is evaluated for multiple 600-seconds trials to avoid distortions.

5.1.1. Performance baseline under CUBIC

Since CUBIC is the default CC protocol in most modern operating systems [46, 47, 48], we conduct multiple 24-hour evaluations to obtain the performance baseline offered by CUBIC in LAN, WAN-1 and WAN-2 (Table 3). These results are also used as performance baselines in following sections.

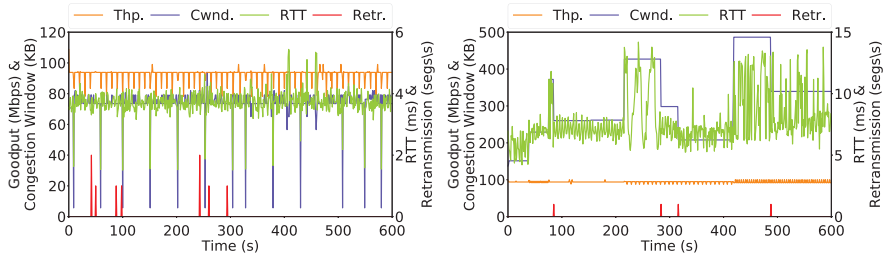
Table 3: The performance baselines in wired scenarios (24-hour evaluation with CUBIC)

Scenario	**Goodput**	**RTT**	**Retransmission**
LAN	93.21Mbps	17.37ms	0.06seg/s
WAN-1	1.07Mbps	36.03ms	9.55seg/s
WAN-2	42.55Mbps	289.12ms	10.91seg/s



(a) LAN (b) WAN-1 (c) WAN-2

Figure 2: The average goodput and 95th percentile RTT of CC protocols involved in wired scenarios



(a) BBR (b) CUBIC

Figure 3: The goodput, RTT, retransmission and congestion window of BBR and CUBIC in LAN

As shown in Table 3, CUBIC fully utilizes the bottleneck bandwidth in LAN and WAN-1. But in WAN-2 (a lossy LFN), CUBIC only achieves about 40% of the bottleneck bandwidth. Such under-utilization of the bottleneck bandwidth is mainly caused by constant packet losses in WAN-2. Detailed analysis on this phenomenon is provided in Section 5.1.4.

5.1.2. LAN

In Figure 2(a), we present the results on average goodput and 95th percentile RTT obtained in LAN. In LAN, most CC protocols achieve at least 90% utilization of bottleneck bandwidth with the 95th percentile RTT around 16ms. In addition, BBR and YeAH achieve similar bottleneck utilization with smaller RTT (i.e., 5ms and 9ms respectively) compared to most CC protocols. Besides, Vegas achieves about 70% bottleneck bandwidth with the lowest RTT (∼2ms).

Figure 3(a) and 3(b) show the running time series of goodput, RTT, con-

gestion window and retransmission for BBR and CUBIC respectively. As Figure 3(a) shows, the RTT and congestion window of BBR remains stable during evaluation. But the RTT of CUBIC fluctuates with its congestion window radically, which is in part due to its loss-based CC scheme. As a loss-based CC protocol, CUBIC probes more available bandwidth by increasing its congestion window until packet lost. But when the bottleneck bandwidth is fully occupied, the extra number of inflight packets[3] congest at bottleneck and create a persistent but unstable queue. Then, the queuing delay is increased (as well as the RTT), and retransmissions occur ultimately if bufferbloat. With the presence of retransmission, CUBIC mitigates the congestion by reducing congestion window (lowering the packet sending rate). Based on these results, an observation can be obtained as below:

Observation-1: BBR fully utilizes the bottleneck bandwidth with lower and more stable RTT than CUBIC in LAN.

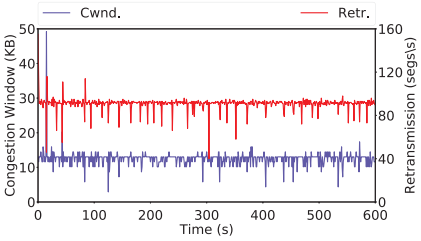
5.1.3. WAN-1

Figure 2(b) illustrates the average goodput and 95th percentile RTT of all CC protocols investigated in WAN-1. In Figure 2(b), all CC protocols present full utilization on the bottleneck bandwidth and maintain similar latency. But the retransmission of BBR is about 90 segments per second (seg/s) on average (Figure 4(a)), and the retransmission of Scalable (Figure 4(b)) and YeAH (Figure 4(c)) are around 42seg/s. As a contrast, the retransmission of other CC protocols, e.g., CUBIC is about 9seg/s on average (Figure 4(d)).

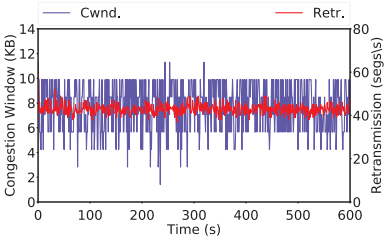
5.1.4. WAN-2

In Figure 2(c), the average goodput and 95th percentile RTT of all CC protocols investigated upon WAN-2 are presented. Based on these results, we can easily find that the RTT of most CC protocols are varying from 245ms to 260ms, and their goodput vary from 30Mbps to 60Mbps. It is worth noting that,

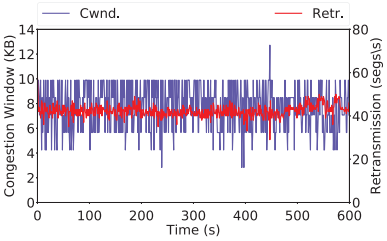
[3]Extra means the packets that exceed the current BDP of underlying flow path.



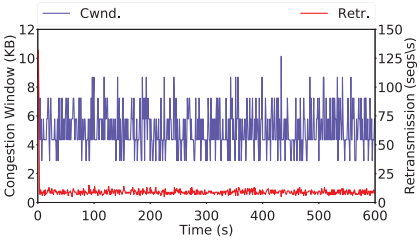
(a) BBR: Retransmission=90.81seg/s



(b) Scalable: Retransmission=42.93seg/s

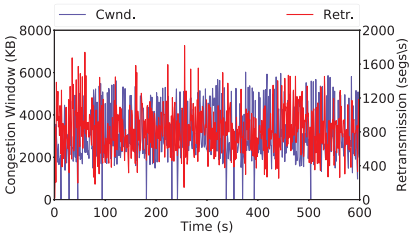


(c) YeAH: Retransmission=42.32seg/s

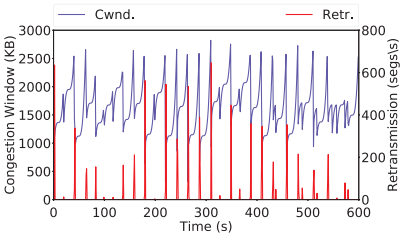


(d) CUBIC: Retransmission=9.47seg/s

Figure 4: The congestion window and retransmission of BBR and CUBIC in WAN-1



(a) BBR: Retransmission=746.84seg/s



(b) CUBIC: Retransmission=14.55seg/s

Figure 5: The congestion window and retransmission of CUBIC and BBR in WAN-2

BBR achieves the highest goodput (∼80Mbps) among these CC protocols with a medium RTT (∼252ms). However, BBR introduces massive retransmissions into the network (∼740seg/s), which is much higher than other CC protocols (e.g., CUBIC).

Figure 5 presents the running histories of BBR and CUBIC. From Figure 5(a), BBR introduces massive retransmissions for more than 700seg/s into the network. Such massive retransmission introduced by BBR is partly due to the implementation integrated in Linux kernel 4.9 that takes no account of packet loss. From Figure 5(b), the retransmission of CUBIC is with burstiness,

and at least one order of magnitude smaller compared to BBR. But retransmissions lead to constant reductions on the congestion window of CUBIC. Thus, CUBIC can not achieve goodput as high as BBR in lossy LFN (e.g., WAN-2). Since the retransmission reported by iperf3 is a mix of retransmission caused by both fast retransmission (duplicate ACKs) and Retransmission TimeOut (RTO), we analyze the packet trace of aforementioned evaluations, and no retransmission caused by RTO is reported.

Based on these results, two observations are presented as below:

Observation-2: CUBIC under-utilizes the bottleneck bandwidth in lossy LFN due to constant packet losses.

Observation-3: BBR achieves the highest goodput among all investigated CC protocols in lossy LFN at the cost of massive retransmissions.

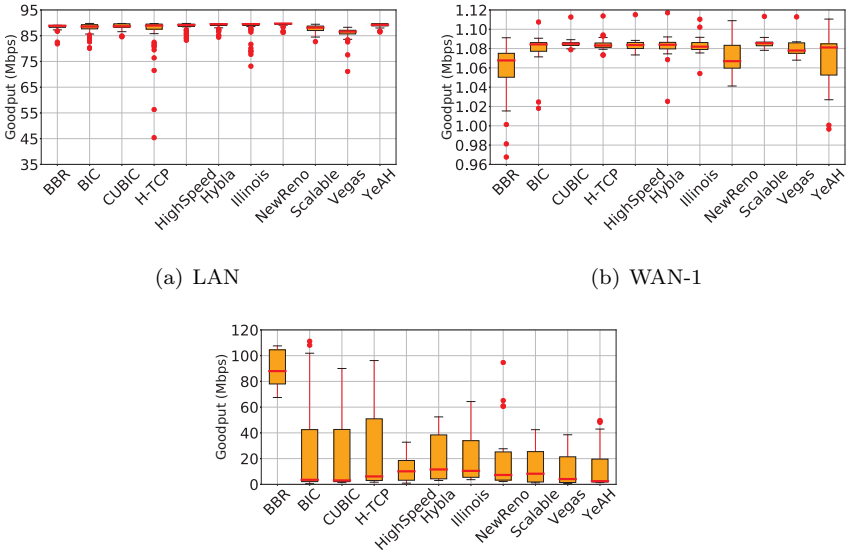
5.2. Large file transfer evaluation

In this section, we use curl to download a large file from remote server and repeat it for 50 times. For LAN and WAN-2 with 100Mbps bottleneck bandwidth, a 1GB file is provided. For WAN-1 with 1Mbps bottleneck bandwidth, a 56MB file is provided. Figure 6 shows the results on average download speed (i.e., goodput) in box charts with median value highlighted.

Figure 6(a) and 6(b) show that all CC protocols are able to fully utilize the bottleneck bandwidth in most cases under LAN and WAN-1. However, the goodput achieved by H-TCP, Illinois and Vegas are more unstable compared with other CC protocols in LAN (Figure 6(a)). In WAN-1, the goodput of BBR, NewReno and YeAH are with less stability compared to other CC protocols from Figure 6(b). But as Figure 6(c) shows, BBR outperforms other CC protocols in WAN-2 with high median value on average download speed (∼90Mbps), while other CC protocols are typically under 15Mbps.

5.3. Friendliness and fairness evaluation

To study friendliness and fairness of CC protocols without background interference, a simulated network is set up for each physical wired scenario in



(a) LAN

(b) WAN-1

(c) WAN-2

Figure 6: Average download speed achieved in wired scenarios

Table 4: Link properties of bottlenecks in all simulated scenarios

Simulated scenarios	**RTT**	**Bottleneck**	**Loss rate**
LAN	2.0ms	100.0Mbps	0.000%
WAN-1	30.0ms	1Mbps	0.001%
WAN-2	260.0ms	100.0Mbps	0.010%

Mininet. See Figure 1(c) for topology. More specifically, the same bottleneck bandwidth, RTT, and the average loss rate of corresponding physical scenario are configured at the bottleneck. Network properties of all simulated scenarios are summaried in Table 4.

To make a comparison, the average goodput of Flow-1 and Flow-2, the utilization on the bottleneck bandwidth (i.e., the sum of their average goodput), and their friendliness and fairness (Jain's Index) are computed[4]. For clarity, the results of CUBIC, BBR, Illinois, Hybla, YeAH and H-TCP are selected in

[4]The CC protocols that achieve at least reasonable friendliness and fairness (Jain's Index$\geq$0.95) are marked in bold.

Table 5: Friendliness vs CUBIC in LAN - Average Goodput (Mbps)

Protocols	**Flow-1**	**CUBIC**	**Sum**	**Jain's Index**
Illinois	17.2	14.8	32.0	**0.99**
H-TCP	6.9	10.1	17.0	**0.97**
Hybla	3.9	11.5	15.4	0.80
YeAH	2.6	18.5	21.1	0.64
BBR	1.0	20.1	21.1	0.55

Table 6: Fairness in LAN - Average Goodput (Mbps)

Protocols	**Flow-1**	**Flow-2**	**Sum**	**Jain's Index**
CUBIC	27.0	30.2	57.2	**0.99**
Illinois	17.9	24.5	42.4	**0.98**
YeAH	4.8	4.0	8.8	**0.99**
Hybla	3.5	3.4	6.9	**0.99**
BBR	4.2	2.7	6.9	**0.95**
H-TCP	2.2	4.5	6.7	0.90

this section, as they achieve higher goodput, friendliness and fairness compared to other CC protocols in most cases. For ease of illustration, full results of friendliness and fairness evaluations against CUBIC and NewReno are provided in Appendix A.

Evaluation results of friendliness and fairness in LAN with 100Mbps bottleneck bandwidth are provided Table 5 and 6. In friendliness evaluation against CUBIC (Table 5), Illinois achieves the highest goodput among all selected CC protocols, and distributes the bottleneck bandwidth with CUBIC fairly. But others do not achieve goodput comparable to Illinois. In fairness evaluation (Table 6), CUBIC achieves the highest utilization (57.2Mbps) among all selected CC protocols with good fairness, followed by Illinois (42.4Mbps). The rest present poor utilization on bottleneck bandwidth ($\leq$10Mbps). Besides, all selected CC protocols except H-TCP present at least reasonable fairness.

In Table 7 and 8, results of friendliness and fairness evaluation based on WAN-1 with 1Mbps bottleneck bandwidth are presented. In WAN-1, none of

Table 7: Friendliness vs CUBIC in WAN-1 - Average Goodput (Kbps)

Protocols	**Flow-1**	**CUBIC**	**Sum**	**Jain's Index**
Illinois	473	151	624	0.79
H-TCP	427	99	526	0.72
Hybla	312	143	455	0.88
YeAH	48	743	791	0.56
BBR	35	863	898	0.54

Table 8: Fairness in WAN-1 - Average Goodput (Kbps)

Protocols	**Flow-1**	**Flow-2**	**Sum**	**Jain's Index**
YeAH	532	429	961	**0.99**
BBR	494	465	959	**0.99**
Hybla	576	116	692	0.69
Illinois	208	483	691	0.86
H-TCP	311	280	591	**0.99**
CUBIC	153	375	528	0.85

the selected CC protocols are able to achieve fair allocation on bottleneck bandwidth in friendliness evaluation against CUBIC (Table 7). In fairness evaluation (Table 8), YeAH and BBR present fair allocation on bottleneck bandwidth with almost full utilization ($\sim$960Kbps). Other CC protocols present poor utilization on bottleneck bandwidth ($\leq$700Kbps), or uneven distribution. By comparing the results from Table 7 and 8, none of the selected CC protocol meets the requirement of high utilization and fair allocation on bottleneck bandwidth in both friendliness evaluation against CUBIC and fairness evaluation.

The results of friendliness and fairness evaluation in WAN-2 with 100Mbps bottleneck bandwidth are presented in Table 9 and 10. From those results, BBR presents high utilization in both friendliness and fairness evaluation ($\geq$90Mbps), while the utilization of other selected CC protocols are all below 12Mbps. In this scenario, Illinois and Hybla can also provide at least 2X improvement in utilization compared to CUBIC. Besides, all selected CC protocols achieve fair allocation on bottleneck bandwidth in fairness evaluation (Table 10).

Table 9: Friendliness vs CUBIC WAN-2 - Average Goodput (Mbps)

Protocols	**Flow-1**	**CUBIC**	**Sum**	**Jain's Index**
BBR	91.01	1.16	92.17	0.51
Illinois	5.49	0.94	6.43	0.67
Hybla	4.70	0.92	5.62	0.69
YeAH	1.65	1.06	2.71	**0.95**
H-TCP	1.59	1.11	2.70	**0.97**

Table 10: Fairness in WAN-2 - Average Goodput (Mbps)

Protocols	**Flow-1**	**Flow-2**	**Sum**	**Jain's Index**
BBR	43.04	48.50	91.54	**0.99**
Illinois	5.54	5.67	11.21	**0.99**
Hybla	4.68	4.55	9.24	**0.99**
H-TCP	1.68	1.31	2.99	**0.98**
YeAH	1.48	1.45	2.93	**0.99**
CUBIC	1.12	0.94	2.06	**0.99**

Since the goodput of BBR is far beyond CUBIC in friendliness evaluation based on WAN-2 (Table 9), we conduct further evaluations under the same configuration to validate our results. And these evaluation present similar results when evaluating BBR (∼90Mbps) or CUBIC (∼1.5Mbps) solely. Also, in Section 5.2, the median goodput of CUBIC is about 1.5Mbps, while the median goodput of BBR is about 90Mbps. These phenomena indicate that BBR is able to achieve more goodput than CUBIC in this scenario. Thus, we believe the link properties (high RTT and loss rate) that limit the goodput of CUBIC in friendliness evaluation.

In network scenario with loss rate, e.g., WAN-2 in this paper, retransmission caused by random error is of frequent occurrence. Since BBR is a BDP-based CC protocol, it will not reduce congestion window as a direct response to retransmission if there is an increasing trend in achieved goodput. However, CUBIC, a loss-based CC protocol, will directly cut down its congestion window in response to retransmission. In Section 5.1.4, we present a more detailed analysis on the

behaviour of CUBIC and BBR in facing retransmission under WAN-2. Furthermore, from [16], higher RTT leads to higher throughput when evaluating BBR, but less throughput when evaluating CUBIC. In this case, we believe high RTT in WAN-2 also contributes to the unfairness of BBR and CUBIC.

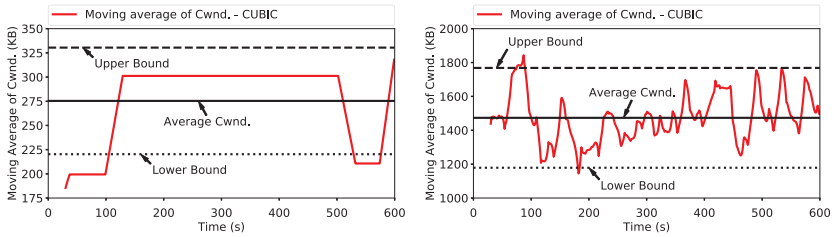
Based on these results, two observations can be made as below:

Observation-4: In LAN, CUBIC achieves the highest utilization in fairness evaluation with optimal share on bottleneck bandwidth.

Observation-5: Hybla and Illinois provide significant improvement in utilization than CUBIC even upon heavily-congested lossy LFN.

5.4. Convergence time and stability

The convergence time t_c (in second) and stability s (in KB) of all CC protocols in LAN, WAN-1 and WAN-2 are presented in Table 11. Results shows that most CC protocols take a considerable amount of time to reach convergence in LAN and WAN-2 with 100Mbps bottleneck bandwidth. But in WAN-1, all investigated CC protocols converge at the beginning of evaluation (t_c close to 30s with $W = 30$). This phenomenon is partially caused by the 1Mbps bottleneck bandwidth, which can be easily achieved full utilization by a small variation in congestion window.



(a) LAN (b) WAN-2

Figure 7: The moving average of congestion window of CUBIC in LAN and WAN-2.

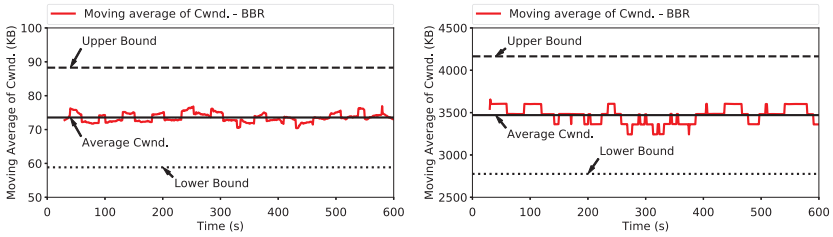
Though LAN and WAN-2 have identical bottleneck bandwidth (100Mbps), the high RTT and loss rate in WAN-2 have different effect on the convergence time and stability of investigated CC protocols. As presented in Table 11, most loss-based CC protocols present faster convergence in WAN-2 (e.g., CUBIC).

Table 11: Dividing all CC protocols into three categories according to the relative size of t_c in LAN and WAN-2

Category	**Protocol**	**LAN**		**WAN-1**		**WAN-2**	
		t_c	s	t_c	s	t_c	s
Faster	BIC	60.4	79	30.0	0.32	30.7	166
	CUBIC	129.3	100	30.0	0.24	53.8	164
	H-TCP	86.1	78	30.0	0.26	53.5	186
	Hybla	95.5	19	30.0	0.34	30.1	164
	Illinois	94.1	31	30.6	0.25	34.5	164
	NewReno	80.3	92	30.1	0.31	55.0	298
	Scalable	41.0	55	30.3	0.37	31.0	275
Similar	BBR	30.0	1	30.0	0.37	32.0	122
	HighSpeed	76.5	152	30.1	0.29	81.9	227
	YeAH	34.9	10	30.4	0.36	37.0	136
Slower	Vegas	45.0	3	30.3	0.36	109.6	192

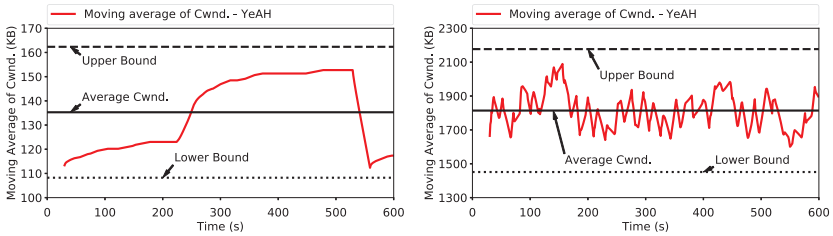
Besides, several CC protocols present similar convergence (e.g., BBR), and Vegas presents slower convergence time in WAN-2. To make a clear demonstration on the effect laid on loss-based CC protocols, the running histories of CUBIC from LAN and WAN-2 are provided in Figure 7. As Figure 7(a) shows, the congestion window of CUBIC is with large plateau after each increase, which leads to a large t_c. This is mainly because of the cubic function that tries to stabilize the data flow between concave growth and convex growth [4]. Figure 7(b) shows, constant retransmissions lead to frequent change on the congestion window, and limit the growth of congestion window in WAN-2. In this case, the convergence time of CUBIC in WAN-2 is reduced to a smaller value than LAN.

Another interesting phenomenon is that BBR and YeAH can almost converge immediately and are more stable than other CC protocols in LAN and WAN-2. Figure 8 presents the running samples of BBR from LAN and WAN-2. In Figure 8(a) and 8(b), the congestion window of BBR fluctuates within a narrow range in both LAN and WAN-2. Since BBR can fully utilize the bottleneck



(a) LAN (b) WAN-2

Figure 8: The moving average of congestion window of BBR in LAN and WAN-2.



(a) LAN (b) WAN-2

Figure 9: The moving average of congestion window of YeAH in LAN and WAN-2.

bandwidth in LAN and WAN-2 (Figure 2), such minor fluctuation in congestion window suggests that it makes accurate estimation on the bottleneck bandwidth. In Figure 9, running samples of YeAH from LAN and WAN-2 are presented. As Figure 9(a) shows, the average congestion window of YeAH (136KB) is almost as twice as BBR (74KB). Since BBR can achieve full utilization on bottleneck bandwidth, such a large congestion window is not necessary. In Figure 9(b), YeAH presents similar fast convergence as BBR. From [30], YeAH employs the loss-based CC schemes of NewReno and Scalable, and decides the functional one by packet delay. Thus, massive retransmissions in WAN-2 also suppress its congestion window and bring faster convergence as other loss-based CC protocols (e.g., CUBIC).

Based on aforementioned results, two observations can be made as below:

Observation-6: Loss-based CC protocols, e.g., CUBIC, have faster convergence and worse stability in lossy LFN.

Observation-7: BBR and YeAH can converge at the beginning of evaluation in LAN/WAN scenarios.

6. Evaluation in wireless LAN

In this section, we evaluate the performance and behaviour of Westwood+ and Veno that specifically designed for wireless LAN, together with CUBIC, NewReno and BBR that has great popularity in wired scenario.

6.1. TCP performance evaluation

To minimize the effect of interference and contention from nearby hosts, selected CC protocols are evaluated with multiple 1-hour tests. Results on goodput (in Mbps), RTT (in ms), retransmission (in segs/s), congestion window (in KB), number of ACK blocks per second[5] (N_{ACK_Block} for short, in block/s), and packets per ACK block (P_{ACK_Block} for short, in packet/block) are presented in Table 12 (for 2.4GHz Wi-Fi network, signal level=-40dBm) and 13 (for 5GHz Wi-Fi network, signal level=-53dBm).

From those results, the performance of CC protocols are greatly improved in 5GHz Wi-Fi scenario compared to 2.4GHz Wi-Fi scenario. For example, the goodput is increased at least 2X, RTT is reduced almost by 60%, and retransmission is reduced by more than 46%. However, in both 2.4GHz and 5GHz scenarios, BBR and CUBIC present some performance issues in wireless LAN. These issues will be deeply investigated in Section 6.1.1 (analysis for BBR) and 6.1.2 (analysis for CUBIC) using data from 2.4GHz wireless scenario as an example.

6.1.1. BBR achieves lower goodput in wireless LAN

In Table 12 and 13, although BBR presents lower RTT and retransmission, its goodput is significantly lower than other CC protocols in wireless scenario. For example, it can be about 30% less compared to CUBIC in 2.4GHz scenario. Considering that BBR achieves even higher goodput compared to CUBIC in WAN-2 (Section 5.1), such performance degradation is mainly due to the complicated interaction between the link characteristics of IEEE 802.11 wireless

[5] A block means one or more continuous acknowledgement(s).

Table 12: Results of performance evaluation in 2.4GHz Wi-Fi network

Protocols	**Thp.**	**RTT**	**Retr.**	**Cwnd.**	N_{ACK_Block}	P_{ACK_Block}
CUBIC	124.7	23.2	0.16	1393	119.15	5.12
NewReno	124.6	17.8	0.10	701	156.58	4.52
BBR	87.6	7.2	0.04	88	347.98	1.21
Westwood+	118.1	18.0	0.07	645	154.98	4.56
Veno	111.1	15.7	0.09	431	159.12	4.83

Table 13: Results of performance evaluation in 5GHz Wi-Fi network

Protocols	**Thp.**	**RTT**	**Retr.**	**Cwnd.**	N_{ACK_Block}	P_{ACK_Block}
CUBIC	357.09	8.61	0.093	1905	416.92	11.17
NewReno	346.68	7.40	0.031	925	426.25	13.67
BBR	276.42	4.44	0.036	256	611.13	6.00
Westwood+	342.16	7.22	0.044	821	440.22	12.33
Veno	338.13	6.91	0.041	607	409.12	13.33

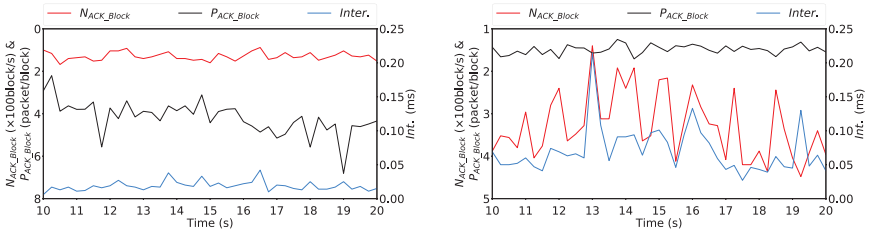
LAN and the congestion control scheme of BBR that dynamically sets the pacing rate of TCP socket.

In *half-duplex*[6] wireless LAN with deactivated RTS/CTS, hosts want to transmit data to each other have to compete for the shared communication medium. As a consequence, contention from nearby hosts is of frequent occurrence. For the purpose of mitigating the effect contention from nearby hosts, we only connect one laptop client to the router, and conduct evaluations at night. In this case, with respect to the wireless NIC of laptop, the contention is mainly from the router forwarding ACKs back to laptop. To find out how contention degrade the performance of BBR, we analyze the packet trace with a focus on two metrics: N_{ACK_Block} and P_{ACK_Block}. The N_{ACK_Block} indicates the switching frequency of transmission direction (due to contention), while P_{ACK_Block} indicates how much data are acknowledged during each continuous

[6]Current IEEE 802.11 standards, e.g., 802.11ac and 802.11n employed in this paper, only support half-duplex communications between hosts [49].

transmission in the link layer.

From Table 12 and 13, the N_{ACK_Block} of BBR is higher than other CC protocols, and the P_{ACK_Block} of BBR is lower than other CC protocols. For instance, N_{ACK_Block} is about 1.75X compared to CUBIC, and P_{ACK_Block} is only about 24% when compared to CUBIC in 2.4GHz wireless scenario. This phenomenon indicates that BBR has more changes in transmission direction, and CUBIC can transmit more TCP data segments each time. Beside, there is also a high correlation between N_{ACK_Block} and the average transmission interval between two TCP data segments[7] (*Inter.* for short) when evaluating BBR. Figure 10 presents the moving histories of N_{ACK_Block}, P_{ACK_Block} and *Inter.* for both CUBIC and BBR during one evaluation (goodput is in a steady state during this time window).



(a) CUBIC (b) BBR

Figure 10: The correlation between N_{ACK_Block}, P_{ACK_Block}, and *Inter.* of CUBIC and BBR

Figure 10(a) shows that the N_{ACK_Block} and *Inter.* of CUBIC is stable. The P_{ACK_Block} of CUBIC is slightly more drastic, but still stays near the average value for most of the time (∼4block/s in this time window). From Figure 10(b), the P_{ACK_Block} of BBR is in a steady state (∼1.5packet/block in this time window). But different from CUBIC, the N_{ACK_Block} and *Inter.* of BBR present strong negative correlation, i.e., the more changes in transmission direction, the smaller interval in sending two continuous data segments. Since P_{ACK_Block} remains near its average value, this phenomenon indicates that the

[7]Data segments are those sent from laptop client to server, carrying data payload to evaluate network performance.

TX process in the link layer is interrupted frequently by incoming ACKs. It is also observed that the *Inter.* of BBR (∼0.07ms) is about 3.5X larger than CUBIC (∼0.02ms).

Deep into the BDP-based congestion control mechanism employed by BBR, it not only sets the value of congestion window (determines how much data to send), but also the value of pacing rate (how fast should data to be sent) of a TCP socket. After entering the *ProbeBW* state where it tries to probe more available bandwidth, it sets the *pacing_gain*[8] to 1.25 to probe in mild way. Since the available data in congestion window is not limited by application (i.e., iperf3) during evaluation, larger *Inter.* indicates that the pacing rate of TCP socket (i.e., the time interval between two continuous packets in TX buffer) calculated by the *send* function of BBR (see Figure 3 in reference [2]) is too small in wireless LAN. Such small pacing rate, i.e., larger TCP segment interval, will allow TCP ACKs forwarded by wireless router to contend the channel with more opportunity, which interrupts continuous data transmission of BBR and reduces its goodput.

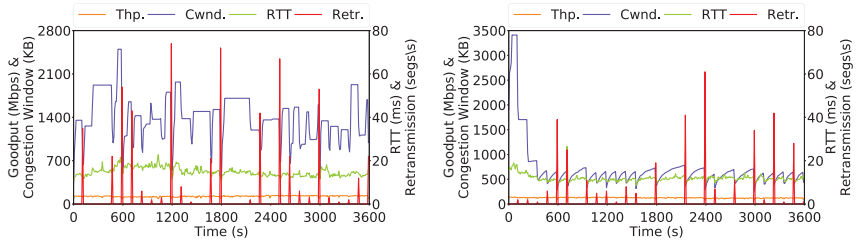
Thus, an observation can be made based on aforementioned materials:

Observation-8: The small pacing rate set by BBR leads to lower goodput in wireless LAN.

6.1.2. CUBIC is more aggressive compared to other CC protocols

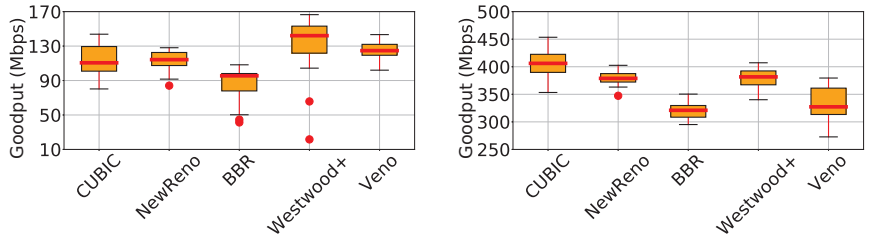
From Table 12 and 13, another interesting phenomenon is that the retransmission and congestion window of CUBIC is nearly as twice as other CC protocols (e.g., NewReno) in both 2.4GHz and 5GHz Wi-Fi scenarios. Figure 11 shows the running samples of CUBIC and NewReno in 2.4GHz Wi-Fi scenario for comparison. As depicted in Figure 11(a), the congestion window of CUBIC is increased drastically after sharp reduction due to the concave profile of the cubic function [4]. Since the retransmissions are mainly caused by network

[8] A parameter that controls how fast the pacing rate is increased. The larger this value is, the faster pacing rate is increased.



(a) CUBIC (b) NewReno

Figure 11: The goodput, congestion window, RTT and retransmission of CUBIC and NewReno in 2.4GHz Wi-Fi scenario



(a) 2.4GHz (b) 5GHz

Figure 12: Large file transfer evaluations in Wi-Fi scenarios

congestion (overflowed buffer), drastic increase in congestion window is easy to cause bufferbloat. Although NewReno also has frequent retransmissions, its congestion window is increased in a linear way, and its pace goes down as RTT increased in Figure 11(b). Most importantly, NewReno achieves almost the same goodput and RTT, but with far less retransmissions and smaller RTT compared to CUBIC (Table 12).

Based on aforementioned analysis, an observation is presented as below:

Observation-9: CUBIC is too aggressive and causes more retransmission than other CC protocols in Wi-Fi networks.

6.2. Large file transfer evaluation

To investigate the goodput of all selected CC protocols when carrying Web traffic, the curl is used to transfer a 3.6GB zip file in this section. For each CC protocol, file transfer operation is repeated for 20 times. The results on average download speed (i.e., goodput) obtained under 2.4GHz (Figure 12(a)) and 5GHz

Table 14: Friendliness vs CUBIC in 2.4GHz Wi-Fi network - Average Goodput (Mbps)

Protocols	**Flow-1**	**CUBIC**	**Sum**	**Jain's Index**
NewReno	13.07	61.23	74.30	0.704
BBR	4.64	51.88	56.53	0.588
Westwood+	14.37	37.00	51.22	0.834
Veno	16.95	49.64	66.60	0.805

Table 15: Friendliness vs CUBIC in 5GHz Wi-Fi network - Average Goodput (Mbps)

Protocols	**Flow-1**	**CUBIC**	**Sum**	**Jain's Index**
NewReno	112.20	130.65	242.86	**0.994**
BBR	38.42	172.83	211.26	0.711
Westwood+	102.03	125.92	227.95	**0.989**
Veno	85.76	139.15	224.92	0.946

Table 16: Fairness in 2.4GHz Wi-Fi network - Average Goodput (Mbps)

Protocols	**Flow-1**	**Flow-2**	**Sum**	**Jain's Index**
CUBIC	55.39	36.75	92.15	**0.960**
NewReno	32.70	41.55	74.25	**0.985**
BBR	90.68	10.12	100.80	0.610
Westwood+	32.81	51.81	84.63	**0.952**
Veno	12.32	51.62	63.95	0.725

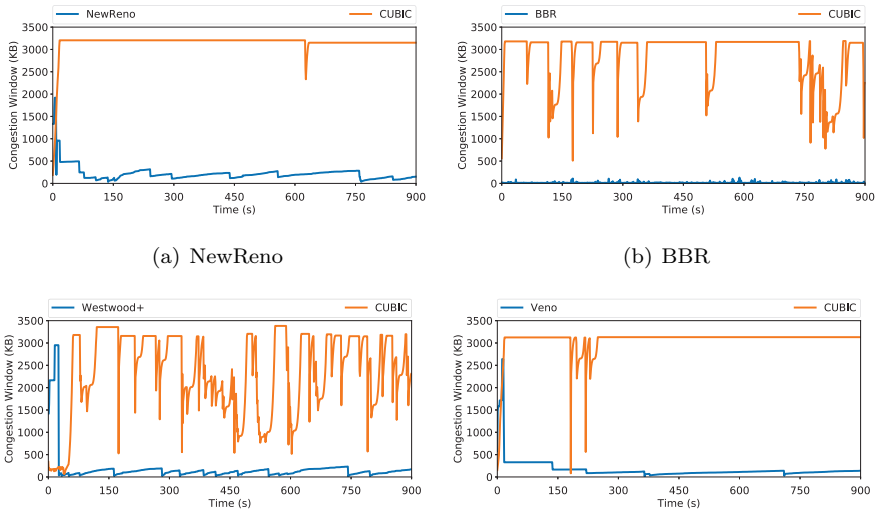
(Figure 12(b)) Wi-Fi scenarios are expressed in box charts with median values highlighted. The median value of BBR is the lowest among all investigated CC protocols in wireless scenario. Besides, the goodput of BBR and Westwood+ is not as stable as others in 2.4GHz Wi-Fi scenario (Figure 12(a)).

6.3. Friendliness and fairness

This section evaluates the friendliness and fairness of all selected CC protocols in physical network with multiple 15min trials. The evaluation setups are as the same as Section 6.1 and 6.2, but Flow-2 (CUBIC) is added as the competing flow (Figure 1(b)). Results on goodput achieved by Flow-1 and Flow-2 (CUBIC), as well as utilization on bottleneck bandwidth and Jain's Index are

Table 17: Fairness in 5GHz Wi-Fi network - Average Goodput (Mbps)

Protocols	**Flow-1**	**Flow-2**	**Sum**	**Jain's Index**
CUBIC	137.67	128.60	266.27	**0.998**
NewReno	100.92	133.65	234.58	**0.980**
BBR	180.75	11.85	192.60	0.565
Westwood+	126.08	95.89	221.97	**0.981**
Veno	55.65	168.72	224.37	0.797



(a) NewReno (b) BBR

(c) Westwood+ (d) Veno

Figure 13: The congestion window of NewReno, BBR, Westwood+ and Veno in friendliness evaluation under 2.4GHz Wi-Fi scenario

presented for friendliness and fairness evaluation in 2.4GHz (Table 14 and 15) and 5GHz (Table 16 and 17) Wi-Fi scenarios.

As presented in Table 14, CUBIC greatly suppresses the goodput of other CC protocols (Flow-1) in 2.4GHz Wi-Fi scenario. To show it clearly, the running samples for NewReno, BBR, Westwood+ and Veno in 2.4GHz Wi-Fi scenario are presented in Figure 13. The congestion window of CUBIC is significantly larger than its competitors, suggesting that CUBIC is too aggressive in 2.4GHz Wi-Fi scenario. But in 5GHz Wi-Fi scenario, NewReno and Westwood+ achieve reasonable friendliness in evaluation against CUBIC (Table 15). Besides, the

friendliness of Veno and BBR are improved in 5GHz scenario. From Table 16 and 17, except for BBR and Veno, all CC protocols achieve at least reasonable distribution on bottleneck in friendliness and fairness evaluation in 5GHz Wi-Fi network.

Based on aforementioned analysis, an observation can be made as below:

Observation-10: CUBIC has poor friendliness in 2.4GHz Wi-Fi network.

6.4. Convergence time and stability

In this section, the convergence time and stability of CC protocols are evaluated in wireless LAN. Table 18 presents the convergence time t_c (in second) and stability s (in KB) of all investigated CC protocols in 2.4GHz and 5GHz Wi-Fi scenarios.

Table 18: Convergence time and stability in Wi-Fi scenarios ($W = 30$)

Protocols	**2.4GHz**		**5GHz**	
	t_c	s	t_c	s
CUBIC	40.0	314	49.0	426
NewReno	364.0	89	125.0	127
BBR	30.0	2	30.0	2
Westwood+	135.4	129	84.0	106
Veno	595.0	60	257.0	90

From Table 18, the convergence time of CUBIC in 5GHz Wi-Fi scenario (49.0s) is larger than 2.4GHz Wi-Fi scenario (40.0s). Also, CUBIC presents worse stability in 5GHz Wi-Fi scenario. Besides, BBR converges at the beginning of evaluation in both 2.4GHz and 5GHz Wi-Fi scenario. Furthermore, NewReno, Westwood+ and Veno present faster convergence in 5GHz Wi-Fi scenario (at least 37% smaller). Based on Table 12 and 13, the RTT in 5GHz Wi-Fi network is significantly reduced compared to 2.4GHz Wi-Fi network. Thus, NewReno, Westwood+ and Veno can reach convergence faster as the ACKs arrive more quickly in 5GHz Wi-Fi scenario.

Based on aforementioned analysis, an observation is presented as below:

Observation-11: NewReno, Westwood+ and Veno present faster convergence in 5GHz Wi-Fi network than 2.4GHz Wi-Fi network.

7. Comparative studies under varied scenarios

In this section, several special issues indirectly affect the performance and behaviour of CC protocols are carefully analyzed. These issues include transmission cost (Section 7.1), convergence in different scenarios (Section 7.2), the effect of congested reverse path (Section 7.3), and the effect of different bottleneck queue size (Section 7.4).

7.1. The transmission cost of BBR and CUBIC

To the best of our knowledge, the transmission cost of BBR has not been studied in previous works. In this section, we investigate the cost diversity of BBR and CUBIC in scenarios with different combination of RTT and loss rate.

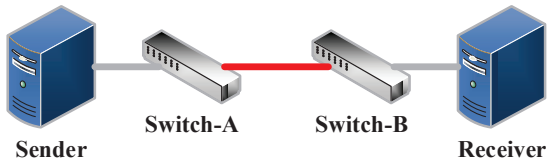


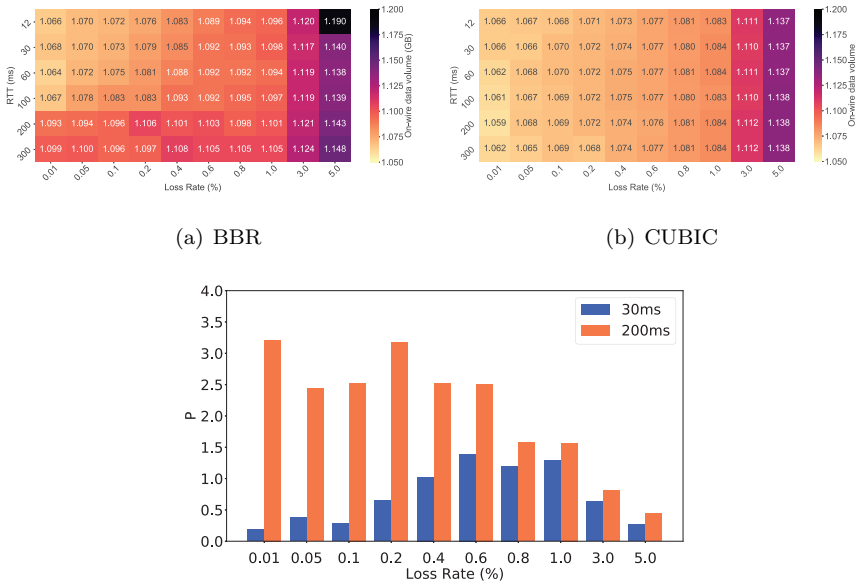
Figure 14: Single bottleneck topology with bottleneck colored in red

We set up a single dumbbell topology composed by two end hosts, and two switches in Mininet as Figure 14, to evaluate *the on-wire data volume*[9], *the average retransmission per packet*[10], and *the average retransmission per loss*[11] of BBR and CUBIC. In this topology, the grey links are with 1Gbps bandwidth capacity, and the red link (bottleneck) is with 100Mbps bandwidth capacity. To simulate different scenarios, different RTT and loss rates are configured at the bottleneck. More specifically, the RTT is set as [10,30, 60, 100, 200, 300]ms,

[9] On-wire data volume is measured by the size of Ethernet frames sent by both sides.

[10] "Packet" stands for all the packets transmitted by the sender.

[11] Packet loss is identified with sender-side packet transmission histories captured by tcpdump.



(a) BBR

(b) CUBIC

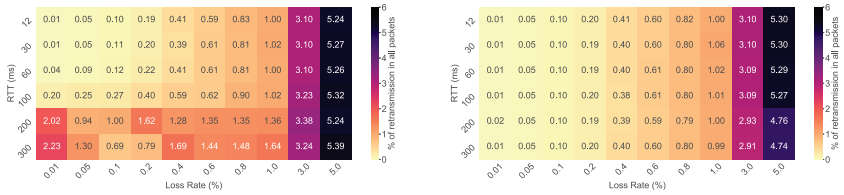
(c) $(V_{BBR} - V_{CUBIC})/V_{CUBIC}$ in percentage

Figure 15: The on-wire data volume of BBR and CUBIC, and the comparison of result obtained when RTT=30ms and RTT=200ms

and loss rate is set as [0.01, 0.05, 0.1, 0.2, 0.4, 0.6, 0.8, 1.0, 3.0, 5.0]%. All combinations of different RTT and loss rates are evaluated with multiple trails that transmitting 1GB data.

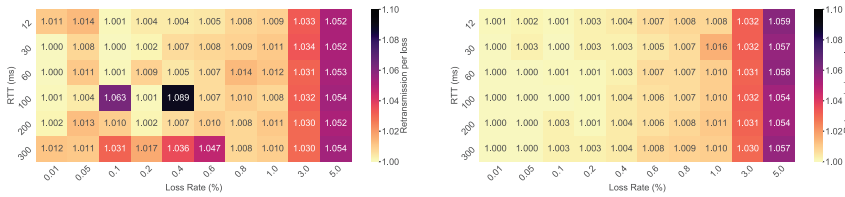
The on-wire data volume of BBR V_{BBR} (in GB) and CUBIC V_{CUBIC} (in GB) under different combinations of RTT and loss rate in Figure 15(a) and 15(b). Results show that V_{BBR} is always larger than V_{CUBIC}. Besides, V_{BBR} from scenarios with large RTT ([200, 300]ms) is increased compared to results from scenarios with small RTT ($\leq$100ms) as presented in Figure 15(a). For instance, V_{BBR} is increased by 2.37% between results obtained from RTT=100ms and RTT=200ms at loss rate=0.01%. Thus, results in Figure 15(a) and 15(b) are all divided into two groups according to network RTT: 1) small RTT ([12, 30, 60, 100]ms), and 2) large RTT ([200, 300]ms). For clarity, results from 30ms and 200ms are used as the example to compare V_{BBR} and V_{CUBIC} in scenario with small and large RTT.

In Figure 15(c), $(V_{BBR} - V_{CUBIC})/V_{CUBIC}$ for results from RTT=30ms and



(a) BBR (b) CUBIC

Figure 16: Heatmap for retransmission per sent packet of BBR and CUBIC



(a) BBR (b) CUBIC

Figure 17: Heatmap for retransmission per loss of BBR and CUBIC

RTT=200ms are computed and expressed in percentage (P for short). When RTT is small (e.g., 30ms), P reaches its peak (1.39%) at loss rate=0.6%, and it is declined in larger loss rates. When the RTT is large (e.g., 200ms), P becomes smaller in most test cases as the loss rate increased (the maximum value is 3.21% at loss rate=0.01%), indicating that V_{BBR} is closing to V_{CUBIC} with increased loss rate. Since the data to be transmitted is identical (1GB), different trends in P suggest that BBR introduces more retransmission in network with large RTT (e.g., 200ms and 300ms) and low loss rate (e.g., $\leq$0.6%).

In Figure 16, results on the average retransmission per packet of BBR (RP_{BBR}) and CUBIC (RP_{CUBIC}) are presented. From Figure 16, RP_{BBR} and RP_{CUBIC} are raised drastically as the loss rate increases. Besides, RTT is with significant effect on RP_{BBR} as presented in Figure 16(a). For example, RP_{BBR} is increased by more than 10X when RTT=200ms than RTT=100ms at loss rate=0.01%. But the increase in RTT presents almost no effect on RP_{CUBIC} as depicted in Figure 16(b). In Figure 17(a) and 17(b), the results on average retransmission per loss of BBR (RL_{BBR}) and CUBIC (RL_{CUBIC}) from different combinations of RTT and loss rates are presented. Based on Figure 17(a), when

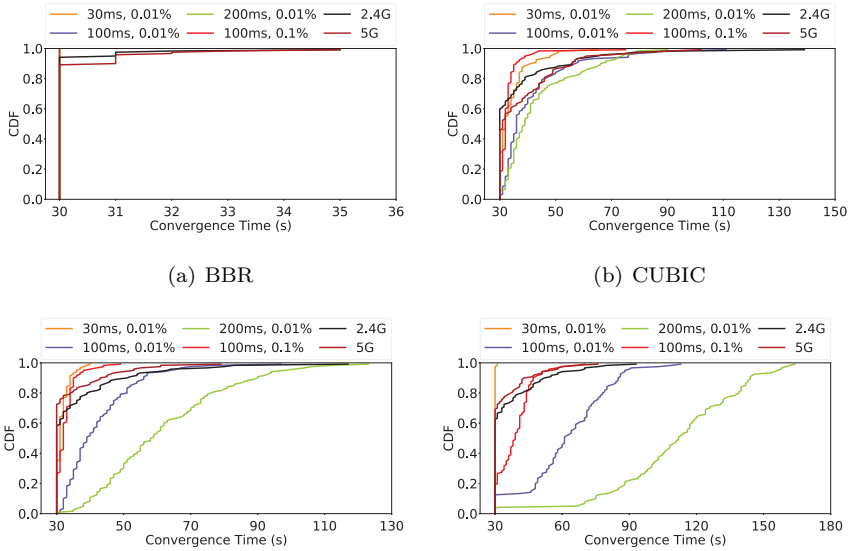
loss rate<3.0%, RL_{BBR} is increased with RTT and loss rate in most cases. But when loss rate≥3.0%, increasing RTT has to no significant effect on RL_{BBR}. Furthermore, RL_{CUBIC} is mainly increased with increased loss rate (i.e., RTT presents no significant effect on RL_{CUBIC}) according to Figure 17(b).

7.2. The convergence time in different scenarios

In this section, the convergence time of BBR, CUBIC, Veno and Westwood+ are evaluated in simulated wired Ethernet LAN and wireless LAN (2.4GHz and 5GHz Wi-Fi networks) with multiple 600s trials. The topology of simulated wired scenarios is identical to Figure 14. In simulated wired scenarios, RTT is set as [30, 100, 200]ms, and loss rate is set as [0.01, 0.10, 1.00]%. Results from RTT=[30, 200]ms at loss rate=0.01% are used to demonstrate the effect of increasing RTT, and results from RTT=100ms and loss rate=[0.01,0.10]% are used to demonstrate the effect of increasing loss rate. The topology of simulated wireless LAN is set as Figure 18 in Mininet-WiFi [50]. In simulated wireless LAN, sender is connected to AP via 2.4GHz (IEEE 802.11n) or 5GHz (IEEE 802.11ac) wireless connection, and receiver is connected to AP via 10Gbps Ethernet link to effectively create a bottleneck at the wireless connection.

Figure 18: Wireless topology simulated by Mininet-WiFi

In Figure 19, convergence time of BBR, CUBIC, Westwood+ and Veno in both simulated wired and wireless scenarios are presented. From Figure 19(a), BBR converges at the beginning of evaluation in most cases (≥90%) upon both simulated wired and wireless scenarios. For CUBIC, increased RTT leads to a significant larger convergence time according to Figure 19(b). For instance, average t_c is 34.1s (RTT=30ms) and 43.5s (RTT=200ms) at loss rate=0.01%. Besides, the increase in RTT also leads to significant larger t_c when using Veno (Figure 19(c)) and Westwood+ (Figure 19(d)). For example, the average t_c for Westwood+ is 30.1s (RTT=30ms) and 109.8s (RTT=200ms) at loss



(a) BBR (b) CUBIC

(c) Veno (d) Westwood+

Figure 19: The convergence time of BBR, CUBIC, Veno and Westwood+ in simulated wired and wireless scenarios ($W = 30$)

rate=0.01%. Besides, Westwood+ converges at the beginning of evaluation when RTT=30ms and loss rate=0.01% in most cases (≥95%). Additionally, increased loss rate leads to smaller t_c for CUBIC, Westwood+ and Veno. For example, the average t_c of CUBIC is 41.1s (loss rate=0.01%) and 32.9s (loss rate=0.1%) at RTT=100ms (Figure 19(a)). In wireless scenarios, Westwood+ and Veno present at least 6.3% smaller t_c in 5GHz Wi-Fi scenario, while CUBIC presents 4.2% smaller t_c in 2.4GHz Wi-Fi scenario.

7.3. The effect of congested reverse path

In TCP connection with heavily-congested reverse path, the goodput of CC protocols which consider delay could be degraded. To reveal the effect of heavily-congested reverse path, the goodput, RTT and retransmission of delay-based CC protocols (e.g., Vegas) and loss-delay-based CC protocol (i.e., Illinois), together with CC protocols with great popularity (i.e., BBR and CUBIC) are evaluated under scenario with heavily-congested reverse path in this section. These evaluations are based on a dumbbell topology with 100Mbps bottleneck bandwidth

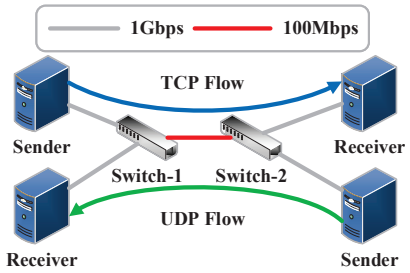


Figure 20: Dumbbell topology used in reverse path evaluation

(Figure 20). The TCP sender and receiver are deployed at the forward path (i.e., TCP Flow in Figure 20). In order to create a heavily-congested reverse path while avoiding occupying the bandwidth capacity of forward path, a UDP flow transmitting data at 100Mbps is deployed at the reverse path (i.e., the UDP Flow in Figure 20). Besides, the round trip propagation delay of forward path and reverse path are all set as 15ms. The loss ratio of all links in this topology is set as zero. Finally, the evaluation is conducted as two separate parts with multiple identical 60-seconds trials: UDP Flow deactivated, and UDP Flow activated. Table 19 reports the results on goodput, RTT and retransmission (Retr.) obtained when the UDP Flow is deactivated and activated. The results are ordered by the goodput achieved by TCP Flow when the UDP Flow is activated.

From Table 19, all investigated CC protocols are able to fully utilize the bottleneck bandwidth (∼90Mbps) in evaluations without congested reverse path. But the RTT of all investigated CC protocols is greatly increased, e.g., the RTT of CUBIC (∼113ms) is about 7.5X compared to configured round trip propagation delay (15ms). Among these CC protocols, BBR (∼30ms) and Vegas (∼20ms) are able to provide a better control on RTT compared to other investigated CC protocols. Moreover, BBR, YeAH and Vegas present almost no retransmission during evaluation without heavily-congested reverse path (<1seg/s).

In evaluation with heavily-congested reverse path (Table 19), most of evaluated CC protocol presents poor utilization (<70Mbps) on bottleneck band-

Table 19: The goodput RTT and retransmission (Retr.) of TCP Flow evaluated when UDP Flow is deactivated or activated

Protocols	**UDP Flow Deactivated**			**UDP Flow Activated**		
	Goodput	**RTT**	**Retr.**	**Goodput**	**RTT**	**Retr.**
CUBIC	91.53	112.92	2.37	90.72	143.17	4.81
H-TCP	90.89	86.97	8.22	82.88	139.32	9.59
BBR	88.59	31.07	0.08	70.54	165.05	47.32
Hybla	91.10	83.29	10.66	66.02	139.22	11.03
Veno	90.05	56.85	7.24	40.82	137.84	7.66
Illinois	90.97	63.82	3.44	30.33	133.92	5.55
Westwood+	90.65	79.32	7.98	29.15	116.77	8.05
YeAH	90.44	53.21	0.05	16.57	121.68	0.14
Vegas	90.96	20.98	0.01	11.35	125.68	0.05

width (100Mbps) with the presence of heavily-congested reverse path. Besides, the RTT of all investigated CC protocols are greatly increased compared to configured value (at least 8X). Since CUBIC is a loss-based CC protocol, it is still be able to fully utilize the bottleneck though its RTT is greatly increased. H-TCP is also be able to achieve high utilization in this scenario as it makes the growth of congestion window mainly invariant to the change in RTT (serves as a scaling factor). It is also observed that the retransmission of BBR also greatly increased with RTT while its goodput is reduced about 20%. However, the goodput of delay-based CC protocols is reduced to varying degrees since the increase in RTT is considered as the signal of congestion.

7.4. The effect of different bottleneck queue size

In this section, the effect of different bottleneck queue size is studied considering the network topology shown in Figure 14. The queue size of bottleneck link ranges from 10 to 1000 packets. Since Mininet does not provide a interface to set the queue size of nodes directly, we use the parameter "max_queue_size" of Mininet link to implement different bottleneck queue size in this topology. By assigning different values to "max_queue_size", we can alter the number of

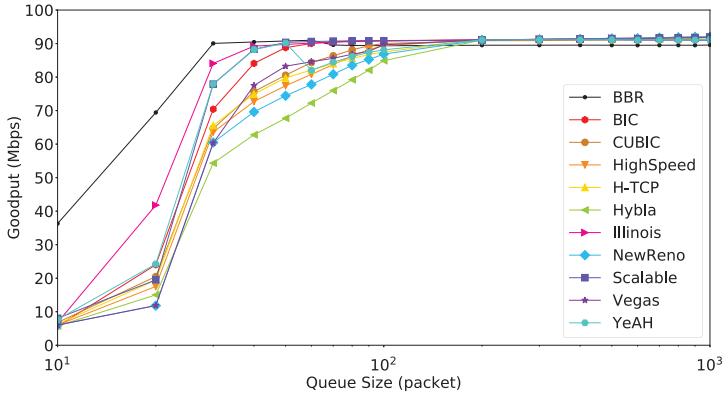


Figure 21: Goodput in network scenarios with different bottleneck queue size

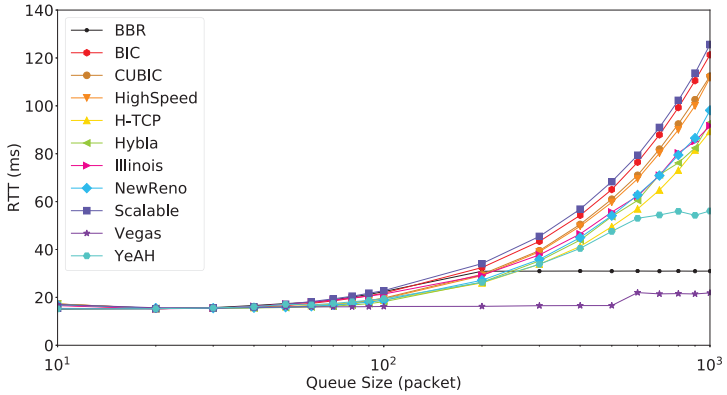


Figure 22: RTT in network scenarios with different bottleneck queue size

packets that can be scheduled upon this link, efficiently simulating the effect of different bottleneck queue size[12]. To avoid retransmission caused by random error, the loss rate of all links is set as 0. Finally, the bottleneck bandwidth is set to 100Mbps, and the round trip propagation delay between sender and receiver is set to 15ms.

Results of goodput are illustrated in Figure 21. When the bottleneck queue size is smaller than 100 packets, the goodput of all investigated CC protocols are increased with bottleneck queue size. When the bottleneck queue size is larger

[12]Packets exceed the length of queue size will be simply dropped according to FIFO (First In First Out) queuing discipline.

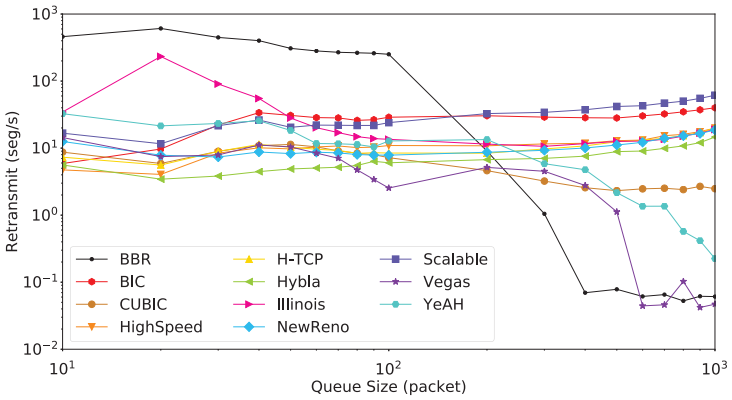


Figure 23: Retransmission in network scenarios with different bottleneck queue size

than 100 packets, their goodput is with minor variation. More specifically, BBR achieves almost full utilization on bottleneck bandwidth in almost all configured bottleneck queue size (≥20packets), while other CC protocols (e.g., CUBIC) achieve full utilization with much larger bottleneck queue size (≥100packets). This is because BBR measures the current BDP dynamically (includes the buffer size of intermediate nodes), and sets a proper congestion window base on its estimation. However, when bottleneck queue is small (≤100packets), other CC protocols are tend to increase the size of congestion window until bufferbloat (causing packet losses and subsequent retransmission), leading to reduced goodput.

In Figure 22, we present the results on the RTT of evaluated CC protocols. From Figure 22, the RTT of most investigated CC protocols are greatly increased when bottleneck queue size is larger than 100 packets, as more packets are delayed in bottleneck queue. With respect to BBR and Vegas, they provide better control (stabilized below 40ms) on RTT compared to other CC protocols (varying from 40ms to more than 100ms as queue size increased). Besides, we also present the results on retransmission in Figure 23. From Figure 23, the retransmission of most investigated CC protocols is increased with bottleneck queue size. However, BBR and Illinois present excessive retransmission compared to other CC protocols when the queue size is smaller than 100 packets.

When the queue size exceeds 100 packets BBR, Vegas and YeAH present lower retransmission compared to other CC protocols.

8. Conclusions

In this paper, we conduct an extensive evaluation on those commonly-used TCP CC protocols, including NewReno, HighSpeed, Scalable, BIC, CUBIC, Vegas, Hybla, YeAH, Westwood+, Veno, H-TCP, Illinois and BBR, upon Ethernet LAN/WAN, wireless LAN and network simulator.

As the default CC protocol, CUBIC fully utilizes the bottleneck bandwidth in all evaluated scenarios except loss LFN. Besides, CUBIC presents the highest utilization on bottleneck bandwidth among all evaluated CC protocols with optimal fairness in LAN. Also, the goodput of CUBIC is almost not affected by heavily-congested reverse path. But in wireless scenarios, CUBIC introduces excessive retransmissions, and presents severe aggressiveness when competing with other CC protocols. As an alternative, BBR full utilizes the bottleneck bandwidth in wired scenarios but with massive retransmissions in lossy LFN. In evaluation with small queue size, BBR can provide the highest goodput among all evaluated CC protocols. However, the goodput of BBR is at least 20% lower than other CC protocols in wireless scenarios due to small pacing rate. In heavily-congested lossy LFN, Illinois and Hybla provide at least 2X improvement in goodput compared to CUBIC. Considering the convergence time, most CC protocols present smaller convergence time in lossy LFN. In wireless LAN, NewReno, Westwood+ and Veno present smaller convergence time in 5GHz Wi-Fi scenario, while CUBIC presents smaller convergence time in 2.4GHz Wi-Fi scenario. Besides, BBR converges at the beginning of evaluation in both wired and wireless scenarios.

In summary, none of the evaluated CC protocol provides optimal performance in all included scenarios, making a CC mechanism that dynamically sets an optimal CC protocol according to recent network properties is urgently needed. Also, since the goodput of BBR is significantly lower than other CC

protocols in wireless LAN, we suggest a thorough revision on its congestion control mechanism to increase goodput while still maintaining low queuing latency. Furthermore, based on our evaluation results obtained from wireless LAN, complicated interaction between TCP-layer and MAC-layer could cause potential performance problem to TCP CC protocol, presenting an open challenge to the design of new CC protocols for wireless LAN. Finally, considering that only Ethernet LAN/WAN and wireless LAN are included in this paper, adding more scenarios (e.g., satellite network and cellular network) would bring more practicability to the conclusions of this paper.

Acknowledgements

This work is partially supported by Chinese National Research Fund (NSFC) Project No. 61772235 and 61602210; the Fundamental Research Funds for the Central Universities (21617409, 21617408 and 21619404); the UK Engineering and Physical Sciences Research Council (EPSRC) grants EP/P004407/2 and EP/P004024/1; the Science and Technology Planning Project of Guangdong (2015A030401043, 2017A040405029); the Science and Technology Planning Project of Guangzhou (201902010041); the Educational Commission of Guangdong Province (2018KTSCX016).

Appendix A. Friendliness and fairness in LAN/WAN scenarios

In this section, we present the full data results of average throughput, utilization and Jain's Index of all investigated CC protocols in friendliness and fairness evaluation and on LAN/WAN. The results of CC protocols selected in Section 5.3 are underlined.

Appendix A.1. LAN

In Table A.20, A.21 and A.22, results on goodput, utilization and Jain's Index are presented to demonstrate their friendliness (against CUBIC or NewReno) and fairness upon LAN with 100Mbps bottleneck bandwidth. As presented in

Table A.20: Friendliness vs CUBIC in LAN - Average Goodput (Mbps)

Protocols	**Flow-1**	**CUBIC**	**Sum**	**Jain's Index**
Scalable	19.6	4.5	24.1	0.72
Illinois	17.2	14.8	32.0	**0.99**
HighSpeed	11.6	21.5	33.1	0.92
BIC	10.8	9.5	20.3	**1.00**
H-TCP	6.9	10.1	17.0	**0.97**
Hybla	3.9	11.5	15.4	0.80
YeAH	2.6	18.5	21.1	0.64
BBR	1.0	20.1	21.1	0.55
Vegas	0.6	31.6	32.2	0.52

Table A.20, Illinois, BIC and H-TCP achieve at least reasonable friendliness against CUBIC. But the utilization of BIC and H-TCP are far less compared to Illinois. Besides, Scalable achieves the highest goodput in evaluation against CUBIC among all CC protocols with poor friendliness. However, in friendliness evaluation against NewReno (Table A.21), none of the CC protocols can achieve high overall utilization and friendliness simultaneously. Based on Table A.22, all CC protocols except HighSpeed and H-TCP achieve at least reasonable fairness in LAN. Additionally, CUBIC achieves the highest utilization (57.2Mbps) with optimal fairness. Besides, Illinois, NewReno and BIC also achieve both utilization greater than 30Mbps with reasonable fairness. But Vegas, YeAH, Hybla, BBR and H-TCP present very low utilization on bottleneck bandwidth ($\leq$20Mbps).

Appendix A.2. WAN-1

In this section, results from friendliness evaluation against CUBIC (Table A.23) and NewReno (Table A.24), as well as fairness evaluation (Table A.25) of all CC protocols based on WAN-1 with 1Mbps bottleneck bandwidth are presented. From Table A.23, none of them can achieve at least reasonable friendliness against CUBIC, though some of them achieve high goodput (e.g., HighSpeed and BIC). In evaluation against NewReno, Illinois achieves high

Table A.21: Friendliness vs NewReno in LAN - Average Goodput (Mbps)

Protocols	**Flow-1**	**NewReno**	**Sum**	**Jain's Index**
HighSpeed	21.5	6.1	27.6	0.76
H-TCP	11.5	4.5	16.0	0.84
BIC	11.4	3.1	14.5	0.75
Hybla	9.6	18.5	28.1	0.91
Illinois	5.8	2.1	7.9	0.82
Scalable	2.6	4.1	6.7	**0.95**
YeAH	2.4	9.4	11.8	0.74
BBR	1.3	9.3	10.6	0.64
Vegas	0.1	19.0	19.1	0.54

Table A.22: Fairness in LAN - Average Goodput (Mbps)

Protocols	**Flow-1**	**Flow-2**	**Sum**	**Jain's Index**
CUBIC	27.0	30.2	57.2	**0.99**
Illinois	17.9	24.5	42.4	**0.98**
NewReno	19.1	16.8	35.9	**0.99**
BIC	18.9	12.1	31.0	**0.95**
Scalable	11.4	14.5	25.9	**0.99**
HighSpeed	5.4	18.7	24.1	0.77
Vegas	5.2	5.1	10.3	**1.00**
YeAH	4.8	4.0	8.8	**0.99**
Hybla	3.5	3.4	6.9	**1.00**
BBR	4.2	2.7	6.9	**0.95**
H-TCP	2.2	4.5	6.7	0.89

utilization (≥800Kbps) and reasonable fairness (Table A.24). HighSpeed and Hybla also achieve reasonable fairness but with poor utilization (≤700Kbps). In fairness evaluation (Table A.25), YeAH and BBR present optimal fairness with almost full utilization (≥950Kbps), while other CC protocols cannot achieve high utilization (≥800Kbps) and at least reasonable fairness in the same time.

Based on the results from friendliness evaluations (against CUBIC and NewReno)

Table A.23: Friendliness vs CUBIC in WAN-1 - Average Goodput (Kbps)

Protocols	**Flow-1**	**CUBIC**	**Sum**	**Jain's Index**
HighSpeed	552	64	616	0.61
BIC	482	104	586	0.71
Illinois	473	151	624	0.79
H-TCP	427	99	526	0.72
Scalable	357	135	492	0.83
Hybla	312	143	455	0.88
YeAH	48	743	791	0.56
BBR	35	863	898	0.54
Vegas	11	934	946	0.51

Table A.24: Friendliness vs NewReno in WAN-1 - Average Goodput (Kbps)

Protocols	**Flow-1**	**NewReno**	**Sum**	**Jain's Index**
BIC	476	213	689	0.87
H-TCP	446	157	603	0.81
Scalable	397	154	551	0.84
Illinois	356	483	839	**0.98**
HighSpeed	277	405	682	**0.97**
Hybla	274	278	552	**0.99**
YeAH	43	691	734	0.56
BBR	36	854	890	0.54
Vegas	12	944	956	0.51

of LAN (Table A.20 and A.21) and WAN-1 (Table A.23 and A.24), the average goodput of YeAH, BBR and Vegas is greatly suppressed. Thus, in a network with no significant RTT and loss rate (e.g., LAN and WAN-1 in this paper), using YeAH, BBR and Vegas would cause severe degradation on goodput when congested with CUBIC or NewReno flow at the bottleneck.

Appendix A.3. WAN-2

In this section, the results of friendliness evaluation against CUBIC (Table A.26) and NewReno (Table A.27), as well as fairness evaluation (Table A.28)

Table A.25: Fairness in WAN-1 - Average Goodput (Kbps)

Protocols	**Flow-1**	**Flow-2**	**Sum**	**Jain's Index**
YeAH	532	429	961	**0.99**
Vegas	328	632	960	0.91
BBR	494	465	959	**0.99**
HighSpeed	177	549	726	0.79
NewReno	448	256	704	0.93
Hybla	576	116	116	0.69
Illinois	208	483	691	0.86
H-TCP	311	280	591	**0.99**
BIC	155	413	568	0.83
CUBIC	153	375	528	0.85
Scalable	139	185	324	**0.98**

Table A.26: Friendliness vs CUBIC WAN-2 - Average Goodput (Mbps)

Protocols	**Flow-1**	**CUBIC**	**Sum**	**Jain's Index**
BBR	91.01	1.16	92.17	0.51
Illinois	5.49	0.94	6.43	0.67
Hybla	4.70	0.92	5.62	0.69
YeAH	1.65	1.06	2.71	**0.95**
H-TCP	1.59	1.11	2.70	**0.97**
Scalable	1.16	0.93	2.09	**0.99**
BIC	1.06	1.02	2.08	**0.99**
Vegas	0.71	1.15	1.86	0.95
HighSpeed	0.58	1.13	1.71	0.91

based on WAN-2 with 100Mbps bottleneck are presented. In friendliness evaluation against CUBIC (Table A.26), the goodput of BBR is larger than 90Mbps, while others are all below 10Mbps. Besides, the goodput of CUBIC is at around 1Mbps, suggesting it is incompatible with network scenario with high loss rate and RTT (e.g., WAN-2). In friendliness evaluation against NewReno (Table A.27), the goodput achieved by all CC protocols are similar to friendli-

Table A.27: Friendliness vs NewReno in WAN-2 - Average Goodput (Mbps)

Protocols	**Flow-1**	**NewReno**	**Sum**	**Jain's Index**
BBR	90.80	0.64	91.44	0.51
Illinois	5.72	0.67	6.39	0.62
Hybla	4.54	0.76	5.30	0.66
YeAH	1.52	0.78	2.30	0.91
H-TCP	1.30	0.77	2.07	0.94
Scalable	1.14	0.69	1.83	0.94
BIC	1.00	0.70	1.70	**0.97**
Vegas	0.79	0.77	1.56	**0.99**
HighSpeed	0.50	0.74	1.24	**0.96**

Table A.28: Fairness in WAN-2 - Average Goodput (Mbps)

Protocols	**Flow-1**	**Flow-2**	**Sum**	**Jain's Index**
BBR	43.04	48.50	91.54	**0.99**
Illinois	5.54	5.67	11.21	**0.99**
Hybla	4.68	4.55	9.23	**0.99**
H-TCP	1.68	1.31	2.99	**0.98**
Scalable	1.12	1.06	2.18	**0.99**
BIC	0.97	1.02	1.99	**0.99**
YeAH	1.48	1.45	2.93	**0.99**
CUBIC	1.12	0.94	2.06	**0.99**
Vegas	0.76	0.72	1.48	**0.99**
NewReno	0.71	0.68	1.39	**0.99**
HighSpeed	0.53	0.49	1.02	**0.99**

ness evaluation against CUBIC, but NewReno is with even smaller goodput (∼0.7Mbps). In fairness evaluation (Table A.27), almost all CC protocols achieve optimal fairness. Similarly, BBR achieves almost full utilization on bottleneck bandwidth (Table A.28). Additionally, Illinois and Hybla can provide at least 2X improvement in goodput compared to other CC protocols (BBR is not included) based on results presented in Table A.26, Table A.27 and A.28.

References

[1] M. A. Alrshah, M. Othman, B. Ali, Z. M. Hanapi, Comparative study of high-speed Linux TCP variants over high-BDP networks, Journal of Network and Computer Applications 43 (2014) 66–75.

[2] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, V. Jacobson, BBR: Congestion-based congestion control, Queue 14 (5) (2016) 50.

[3] K. R. Fall, W. R. Stevens, TCP/IP illustrated, volume 1: The protocols, Addison-Wesley, 2011.

[4] S. Ha, I. Rhee, L. Xu, CUBIC: a new TCP-friendly high-speed TCP variant, ACM SIGOPS Operating Systems Review 42 (5) (2008) 64–74.

[5] C. Caini, R. Firrincieli, TCP Hybla: a TCP enhancement for heterogeneous networks, International Journal of Satellite Communications and Networking 22 (5) (2004) 547–566.

[6] S. Liu, T. Başar, R. Srikant, Performance Evaluation 65 (6) (2008) 417–440.

[7] L. Xu, K. Harfoush, I. Rhee, Binary Increase Congestion control (BIC) for fast long-distance networks, in: INFOCOM 2004 Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies, Vol. 4, IEEE, 2004, pp. 2514–2524.

[8] S. Floyd, T. Henderson, A. Gurtov, The NewReno modification to TCP's fast recovery algorithm, Tech. rep. (2004).

[9] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, R. Wang, TCP westwood: Bandwidth estimation for enhanced transport over wireless links, in: Proceedings of the 7th annual international conference on Mobile computing and networking, ACM, 2001, pp. 287–297.

[10] C. P. Fu, S. C. Liew, TCP Veno: TCP enhancement for transmission over wireless access networks, IEEE Journal on Selected Areas in Communications 21 (2) (2003) 216–228.

[11] T. Lukaseder, L. Bradatsch, B. Erb, R. W. Van Der Heijden, F. Kargl, A comparison of TCP congestion control algorithms in 10G networks, in: 2016 IEEE 41st Conference on Local Computer Networks (LCN), IEEE, 2016, pp. 706–714.

[12] T. A. N. Nguyen, S. Gangadhar, J. P. Sterbenz, Performance Evaluation of TCP Congestion Control Algorithms in Data Center Networks, in: Proceedings of the 11th International Conference on Future Internet Technologies, ACM, 2016, pp. 21–28.

[13] D. Murray, K. Ong, Experimental evaluation of congestion avoidance, in: 2015 IEEE 29th International Conference on Advanced Information Networking and Applications (AINA), IEEE, 2015, pp. 1–8.

[14] K. Ong, D. Murray, T. McGill, Large-Sample comparison of TCP congestion control mechanisms over wireless networks, in: 2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA), IEEE, 2016, pp. 420–426.

[15] M. Hock, R. Bless, M. Zitterbart, Experimental evaluation of BBR congestion control, in: 2017 IEEE 25th International Conference on Network Protocols (ICNP), IEEE, 2017, pp. 1–10.

[16] D. Scholz, B. Jaeger, L. Schwaighofer, D. Raumer, F. Geyer, G. Carle, Towards a deeper understanding of TCP BBR congestion control, in: 2018 IFIP Networking Conference (IFIP Networking) and Workshops, IEEE, 2019, pp. 1–9.

[17] B. Turkovic, F. A. Kuipers, S. Uhlig, Fifty Shades of Congestion Control: A Performance and Interactions Evaluation, arXiv preprint arXiv:1903.03852.

[18] C. Callegari, S. Giordano, M. Pagano, T. Pepe, Behavior analysis of TCP Linux variants, Computer Networks 56 (1) (2012) 462–476.

[19] H. Alakoca, M. Karaca, G. K. Kurt, Performance of TCP over 802.11 ac based WLANs via testbed measurements, in: 2015 International Sym-

posium on Wireless Communication Systems (ISWCS), IEEE, 2015, pp. 611–615.

[20] L. Li, K. Xu, T. Li, K. Zheng, C. Peng, D. Wang, X. Wang, M. Shen, R. Mijumbi, A measurement study on multi-path TCP with multiple cellular carriers on high speed rails, in: Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication, ACM, 2018, pp. 161–175.

[21] E. Atxutegi, F. Liberal, H. K. Haile, K.-J. Grinnemo, A. Brunstrom, A. Arvidsson, On the use of TCP BBR in cellular networks, IEEE Communications Magazine 56 (3) (2018) 172–179.

[22] F. Li, J. W. Chung, X. Jiang, Driving TCP Congestion Control Algorithms on Highway, Proceedings of Netdev 2.

[23] L. Li, K. Xu, D. Wang, C. Peng, Q. Xiao, R. Mijumbi, A measurement study on TCP behaviors in HSPA+ networks on high-speed rails, in: 2015 IEEE Conference on Computer Communications (INFOCOM), IEEE, 2015, pp. 2731–2739.

[24] B. Nguyen, A. Banerjee, V. Gopalakrishnan, S. Kasera, S. Lee, A. Shaikh, J. Van der Merwe, Towards understanding TCP performance on LTE/EPC mobile networks, in: Proceedings of the 4th workshop on All things cellular: operations, applications, & challenges, ACM, 2014, pp. 41–46.

[25] Grieco, A. Luigi, Mascolo, Saverio, Performance evaluation and comparison of Westwood+, New Reno, and Vegas TCP congestion control, ACM SIGCOMM Computer Communication Review 34 (2) (2004) 25–38.

[26] A. Dell'Aera, L. A. Grieco, S. Mascolo, Linux 2.4 Implementation of Westwood+ TCP with rate-halving: A Performance Evaluation over the Internet, in: 2004 IEEE International Conference on Communications (ICC), Vol. 4, IEEE, 2004, pp. 2092–2096.

[27] S. Floyd, RFC 3649 - HighSpeed TCP for Large Congestion Windows, https://www.ietf.org/rfc/rfc3649.txt, [Online; accessed 10-June-2019].

[28] T. Kelly, Scalable TCP: Improving performance in highspeed wide area networks, ACM SIGCOMM computer communication Review 33 (2) (2003) 83–91.

[29] L. S. Brakmo, L. L. Peterson, TCP Vegas: End to end congestion avoidance on a global Internet, IEEE Journal on Selected Areas in Communications 13 (8) (1995) 1465–1480.

[30] A. Baiocchi, A. P. Castellani, F. Vacirca, YeAH-TCP: Yet Another High-speed TCP, in: Proc. PFLDnet, Vol. 7, 2007, pp. 37–42.

[31] D. Leith, R. Shorten, H-TCP: TCP congestion control for high bandwidth-delay product paths, draft-leith-tcp-htcp-06 (work in progress).

[32] esnet, iperf3: A TCP, UDP, and SCTP network bandwidth measurement tool, https://github.com/esnet/iperf, [Online; accessed 10-June-2019].

[33] curl, A command line tool and library for transferring data with URL syntax, https://github.com/curl/curl, [Online; accessed 10-June-2019].

[34] R. Jain, D. Chiu, W. Hawe, A Quantitative Measure Of Fairness And Discrimination For Resource Allocation In Shared Computer Systems, Computer Science cs.ni/9809099.

[35] Y.-T. Li, D. Leith, R. N. Shorten, Experimental evaluation of TCP protocols for high-speed networks, IEEE/ACM Transactions on Networking 15 (5) (2007) 1109–1122.

[36] Alibaba Cloud, An integrated suite of cloud products, services and solutions, https://www.alibabacloud.com/, [Online; accessed 10-June-2019].

[37] AWS, Elastic Compute Cloud (EC2) Cloud Server Hosting, https://amazonaws-china.com/ec2/, [Online; accessed 10-June-2019].

[38] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, N. McKeown, Reproducible network experiments using container-based emulation, in: Proceedings of the 8th international conference on Emerging networking experiments and technologies, ACM, 2012, pp. 253–264.

[39] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow, TCP selective acknowledgment options, Tech. rep. (1996).

[40] M. Mathis, J. Mahdavi, Forward acknowledgement: Refining TCP congestion control, in: ACM SIGCOMM Computer Communication Review, Vol. 26, ACM, 1996, pp. 281–291.

[41] S. Floyd, J. Mahdavi, M. Mathis, M. Podolsky, An extension to the selective acknowledgement (SACK) option for TCP, Tech. rep. (2000).

[42] A. Kuzmanovic, A. Mondal, S. Floyd, K. Ramakrishnan, Adding Explicit Congestion Notification (ECN) Capability to TCP's SYN/ACK Packets, Tech. rep. (2009).

[43] embeddedsystesting.com, What is RTS threshold in wireless, `https://sites.google.com/site/embeddedsystesting/wireless-protocols-and-basics-of-wireless-protocols-wlan-802-11a-b-g-n/what-is-rts-threshold-in-wireless`, [Online; accessed 10-June-2019].

[44] www.cisco.com, Advanced Radio Settings, `https://www.cisco.com/assets/sol/sb/isa500_emulator/help/guide/ae1269129.html`, [Online; accessed 10-June-2019].

[45] wiki.dd-wrt.com, Advanced Wireless Settings, `https://wiki.dd-wrt.com/wiki/index.php/Advanced_wireless_settings#RTS_Threshold`, [Online; accessed 10-June-2019].

[46] Srikant, Windows 10 19H1 Build 18214 and RS5 build 17335 features and changelog, `https://windows101tricks.com/windows-10-19h1-build-18214-rs5-build-17335-features-and-changelog/`, [Online; accessed 10-June-2019].

[47] L. Torvalds, linux, https://github.com/torvalds/linux/releases/tag/v4.18, [Online; accessed 10-June-2019].

[48] superuser.com, Which congestion control algorithm is used by the TCP stack in OS X?, https://superuser.com/questions/865896/which-congestion-control-algorithm-is-used-by-the-tcp-stack-in-os-x, [Online; accessed 10-June-2019].

[49] J. Kim, I. Lee, 802.11 wlan: history and new enabling mimo techniques for next generation standards, IEEE Communications Magazine 53 (3) (2015) 134–140.

[50] INTRIG, Mininet-WiFi: Emulator for Software-Defined Wireless Networks, https://github.com/intrig-unicamp/mininet-wifi, [Online; accessed 10-June-2019].