Performance Study of Improved BBR Congestion Control Algorithm using QUIC in Wireless LAN

Mengjie Zuo
School of Information Engineering
Wuhan University of Technology
Wuhan, China
zmj@whut.edu.cn

Haoying Wu
School of Information Engineering
Wuhan University of Technology
Wuhan, China
why_dd@whut.edu.cn

Yi Han*
School of Information Engineering
Wuhan University of Technology
Wuhan, China
hanyi@whut.edu.cn

Yi Zhong
School of Information Engineering
Wuhan University of Technology
Wuhan, China
zhongyi@whut.edu.cn

Jiazheng Wang
IT Department
Liuzhou Wuling Automobile Industry Co., Ltd
Liuzhou, China
wangjiazheng@wuling.com.cn

***Abstract*— Congestion control enables network nodes to adopt certain approaches to avoid network congestion as much as possible, and respond when network congestion occurs. BBR is a transmission rate-based congestion control algorithm proposed by Google for Chromium QUIC and Linux kernel deployment. The new transmission protocol QUIC proposed by Google has a better congestion control mechanism than TCP. Based on the existing BBR congestion control algorithm, this paper proposes the IBCCA congestion control algorithm and evaluates the performance of the IBCCA, CUBIC, and BBR algorithms under the QUIC protocol in wireless LAN. When the network traffic burst flow is small or the burst duration is short, IBCCA achieves higher network goodput and lower transmission round-trip time. In the experiment of multi-stream transmission, the IBCCA algorithm achieves higher fairness while the goodput and RTT are at the same level. The BBR algorithm achieves a smaller RTT, and the goodput of the IBCCA algorithm is not much different from the CUBIC algorithm.**

Keywords—congestion control, burst flow, multi-stream, fairness

I. Introduction

The QUIC (Quick UDP Internet connection) protocol was first proposed by Google in 2013 as an application layer data transmission protocol used to replace TCP [1]. The QUIC protocol is based on the UDP protocol and provides reliable, orderly, safe, and fast transmission services. Although QUIC uses the UDP infrastructure, it does not rely on the characteristics of UDP. At the same time, QUIC must establish a connection with the other peer before transmission, A handshake is necessary for the data to be transmitted in advance. QUIC provides connection-oriented end-to-end reliable transmission [2].

For a multi-user network transmission scenario in QUIC, multiple users may request connections simultaneously and share the same bottleneck link, as shown in Fig. 1. There are many concurrent senders in a multi-user network, and the network transmission delay is variable. What's more, these concurrent streams will compete for the limited network resource, and thus resulting in network congestion during the transmission. The packets will eventually line up in the buffer queue, increasing queuing delay and packet loss possibility. QUIC adopts a burst traffic mechanism for web browsing data to optimize the user experience in a short period. The occurrence of such burst traffic will increase the burden of network data transmission, as well as increase the risk of network congestion or make the network more congested, which eventually reduces the network transmission throughput.

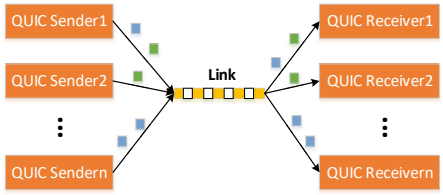


Fig. 1. Dumbbell network topology based on QUIC protocol

To effectively reduce the occurrence of network congestion and stabilize the data flow, many congestion control algorithms have been proposed to control the packet transmission rate. For example, NewReno[3], Vegas[4], BIC[5], CUBIC[6], BBR[7], and many other congestion control algorithms. The current widely used congestion control algorithm is undoubtedly Google's BBR algorithm[7], which uses different methods to try to estimate bandwidth and RTT, and a feedback-driven autonomous adjustment mechanism to keep the initial value of the congestion window consistent with the network capacity.

Inspired by this, this paper proposes the improved BBR congestion control algorithm (IBCCA) and analyses its performance using the QUIC under various network transmission scenarios. In multi-stream and burst traffic scenarios, the IBCCA algorithm is proved to achieve higher fairness, while maintaining a high network performance level.

The rest of this paper is organized as follows. Section 2 introduces related work, and Section 3 describes the BBR congestion control algorithm and the proposed IBCCA algorithm. The algorithm evaluation and experimental results are presented in Section 4. Finally, Section 5 concludes.

II. Related Works

QUIC transmission occurs faster than TCP transmission, which can be attributed to high throughput and effective bandwidth usage. QUIC can also use data packet adjustment to estimate the available bandwidth of the link by tracking the interval between the data packets at the receiving end and the sending end [8].

QUIC is generally more stable than TCP. Even though QUIC uses the same congestion control algorithm as TCP,

978-1-6654-6667-7/21/$31.00 ©2021 IEEE
DOI 10.1109/IUCC-CIT-DSCI-SmartCNS55181.2021.00085

when competing with TCP streams, QUIC still almost always uses more bandwidth than its reasonable bandwidth [10]. However, in terms of user experience of QUIC protocol network transmission, the first study by Rüth et al. [11] found that in slower networks, actual users can usually perceive small differences in technical performance. Compared to TCP, people seem to prefer QUIC. However, if the network speed increases, the difficulty of finding the difference will increase. Minh et al. [12] studied the impact of QUIC and HTTP/2 on scalable video streaming. Experimental results show that the benefits of QUIC are significantly different from our proposed method in the case of packet loss and retransmission. Compared to HTTP/2, it improves the average video quality and provides smoother adaptive behavior. Arisu et al. [13] measured the streaming performance of QUIC on wireless and cellular networks, and found that QUIC leads to faster media streaming start, better streaming, and seeking experience, especially in the case of high network congestion. And it has better performance than TCP when viewers move and switch between wireless networks.

QUIC implements many TCP congestion control algorithms. Brakmo and Peterson's Vegas [4] is the first implementation to use the delay as a congestion signal. The CUBIC algorithm of the congestion control algorithm [6] The increase in the congestion window depends only on the packet loss rate. CUBIC greatly increases the window size, increases the queue and real-time transmission time, and causes buffer expansion or low throughput. Different from the CUBIC protocol, a distributed congestion control protocol BBR proposed by Google [7] attempts to achieve the best operating point by keeping CWND (Congestion Window) equal to BDP (Bandwidth Delay Product) by periodically measuring network capacity [14]. BBR is deployed on Google's B4backbone and compared with CUBIC, the throughput is improved by several orders of magnitude. BBR runs purely on the sender and does not require changes to the protocol, receiver, or network to make it incrementally deployable [7].

In recent years, many researchers have studied the BBR algorithm under the QUIC protocol. Kim et al. [15] proposed a bottleneck queue establishment suppression method, which prevents the establishment of unnecessary persistent queues by limiting the congestion window in the ProbeBW phase-detection bandwidth interval. Compared with the original BBR, it can significantly improve the relationship between different RTT flows. Fairness. Ware et al. [16] found that the flight limit of BBR is the core of BBR's behavior on the Internet. When BBR competes with other traffic (BBR, CUBIC, or Reno, etc.), it sends data packets at a rate determined entirely by its upper limit of flight. Even if a lot of loss-based traffic reaches the network, BBR will not reduce its sending rate. This is the cause of reports arguing that BBR is 'unfair' to legacy TCPs. Zhang et al. [17] aimed at the problem of high packet loss rate and large transmission delay caused by the BBR congestion control algorithm when BBR streams compete for bandwidth resources and proposed a delay response BBR algorithm for real-time video transmission. The results show that, compared with QUIC-BBR and WebRTC-BBR, Delay-BBR can achieve lower transmission delay and lower packet loss rate. In addition, compared with the benchmark algorithm, the packet scheduling algorithm working in conjunction with the rate control algorithm achieves a lower frame transmission delay.

III. Proposed Algorithm

A. BBR Congestion Control Algorithm

The BBR algorithm updates the estimated bottleneck bandwidth (BtlBw) of the available throughput over the network and the estimated baseline round trip time propagation time (Rtprop) to calculate the transmission rate estimated.

BBR has defined the states in different congestion phases: StartUp, Drain, ProbeBW, and ProbeRTT[16]. Their transition process is shown in Fig. 2.

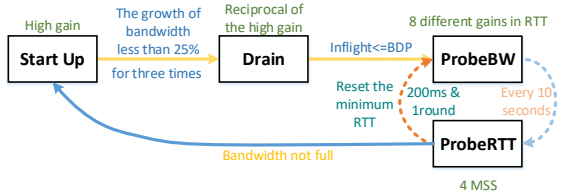


Fig. 2. BBR algorithm congestion control mechanism

When the connection starts, it enters the StartUp state. In the StartUp phase uses high gain to double its transmission rate to detect the maximum available bandwidth, and calculates the pacing rate and congestion window size (cwnd). When it is estimated that the transmission sending rate increment is less than 25% for 3 times consecutively, it is considered to be close to the maximum available bandwidth. The estimated bandwidth stops increasing, and the BBR algorithm enters the Drain state. The Drain state uses the reciprocal of the high gain in the StartUp state to clear the excess queue accumulated during the startup phase. When the data in flight is too large and the pipeline is still overloaded, the Drain state is maintained. When the data packet in flight is less than BDP, the status changes to ProbeBW. The ProbeBW state is stable. In this state, the sending rate is controlled by different gains [1.25, 0.75, 1, 1, 1, 1, 1, 1, 1, 1] in the 8 RTTs. Different from the behaviors in the StartUp state, it uses high gain to expand pacing rate and cwnd. The gain of 1.25 is to detect the bandwidth limit. If there is additional bandwidth available, the sender will increase the sending rate to occupy it. If a small RTT is not observed within 10 seconds, it is determined that the link seems to be in a congested state, and the state is set to ProbeRTT. During ProbeRTT, BBR limits its in-flight data volume to 4 data packets to allow the intermediate router to exhaust the occupied buffer, and ensure that its data packets do not occupy the queue in the network. It measures the RTT of these data packets for at least 200 milliseconds or a regular round trip time of a data packet, then resets the minimum RTT. If ProbeRTT times out and BtlBw is full, BBR returns to ProbeBW state. If ProbeRTT times out and BtlBw is not full, BBR returns to StartUp state.

In a multi-user network transmission scenario, multiple BBR streams use the maximum estimated bandwidth sampled by real-time transmission time, which can easily cause the total transmission rate to exceed the bottleneck capacity, and data packets will be cached in the router. Similarly, when a burst of traffic arrives, the network transmission queue is filled. Both of these conditions will lead to an increase in transmission delay. A smaller RTT is difficult to reappear within 10 seconds, and the detection test state that lasts for up to 200 milliseconds may not be enough for the router cache queue to be emptied[17]. This not only deviates from the maximum estimated bandwidth detected in the previous period but sometimes causes serious congestion. It can only be restarted

by entering the StartUp state. In this case, it will cause unnecessary delays. Moreover, when the burst traffic is more aggressive, BBR does not take the initiative to occupy the queue buffer. The RTT detected during the detection period is low, and the bandwidth estimation tends to be more conservative. Thus, the bandwidth utilization rate is not optimized, resulting in a low convergence speed of the algorithm. If the ProbeRTT state can empty the queue, no measures have been taken to ensure fairness during the process of multiple BBR streams from the ProbeRTT state to the ProbeBW state, so there is no guarantee that the BBR stream after the restoration will remain at an optimal control point to deliver high throughput.

B. Improved BBR Congestion Control Algorithm

Considering the above issues, this paper proposes the improved BBR congestion control algorithm (IBCCA) presented in Algorithm 1.

Algorithm 1 Improved BBR Congestion Control Algorithm

Input: $Blocked$, $delay_i$, I, $BCount_i$.
Output: $Rate_i$, RTT_i.
Define:
I: the set of the total number of packets;
$derate$: the sum of the differences between the Rtprop values in three tests;
$Blocked$: the congestion signal;
$delay_i$: the delay of the i-th packet;
RTT_i: the round-trip time of the i-th packet;

1: Initialize $i \leftarrow 0$, $j \leftarrow i+1$, $tin_i \leftarrow 0$, $Blocked \leftarrow 0$
2: **for** $i \in I, j \in I$ **do**
3: Calculate the value of $derate = \sum_{k=i}^{i+3}(delay_k - delay_{k+1})$
4: **if** $derate >= 0$ **or** $Blocked = 0$ **and** in ProbeBW state **then**
5: keep in ProbeBW state
6: **if** $delay_i - delay_{i+1} < 0$ **then**
7: the detection bandwidth gain ∈ [0.75, 1, 1, 1, 1, 1, 1, 1.25];
8: **end if**
9: **elseif** $derate < 0$ **and** $Blocked = i$ **and** in ProbeBW state **then**
10: change to ProbeRTT state
11: **elseif** $derate < 0$ **and** $Blocked = i$ **and** in ProbeRTT state **then**
12: Calculate the value of $tin_j = tin_{j-1}\left(\frac{delay_i}{delay_{i-1}}\right)^{BCount_i}, j = i+1$
13: increase the time in ProbeRTT state by tin_j milliseconds
14: **end if**
15: calculate $goodput$;
16: **end for**
17: **return** $goodput$, RTT_i

If there are no extra data packets to be processed at the transport layer, or the calculated network delay does not show a significant increase, even if the RTT does not decrease within 10 seconds, the algorithm remains in the ProbeBW state without the need to decrease its transmission rate. In this way, the stream will be able to compete with the aggressive stream during the transmission process. In the ProbeBW state, considering the situation where the network cannot deal with the network congestion due to the sudden traffic burst, the IBCCA transits from ProbeBW to the ProbeRTT when it meets the following two requirements: 1) the transmission layer sends a congestion signal because it cannot process the high volume of data, and 2) the network delay is showing an increasing trend, indicated by the negative sum of the differences between the Rtprop values in three tests. This algorithm can adapt the state to the possible congestion in time when there is burst traffic, and avoid high-speed transmission when the queue is full. IBCCA considers both network transmission performance optimization and fairness among multiple streams. When the network suffers from the transmission traffic congestion caused by burst or multi-user concurrency traffic, the increased time maintained in the ProbeRTT state can allow more time to empty the router buffer queue, to avoid transmission delay increases. This reduces the sending rate, avoids overestimation of BDP, and reduces packet loss and retransmission. Reducing the time between ProbeRTT states will help BBR restore its original stable state when it encounters a burst [18]. This is implemented in IBCCA by enabling the transition to ProbeRTT state in advance to heavy congestion.

According to the design of IBCCA, line 3 to line 8 of Algorithm 1 are to increase the time maintained in the ProbeBW state to better compete with other streams. Line 9 to line 10 are to adapt the state in congestion earlier when there is burst traffic. Line 11 to line 14 allows more time to empty the router's buffer queue when congestion occurs.

In the multi-user scenario, define the set $S = \{1,2,\cdots,L\}$, where L is the total number of streams. The measured throughputs of multiple flows $[t_1, t_2, \cdots, t_L]$ are used to calculate fairness using Jain's index[19], which is often recognized as a fairness metric or the measurement in network engineering to determine whether users or applications have fair sharing of system resources. The calculation is described in (1).

$$Jain(t_1, t_2, \ldots, t_L) = \frac{(\sum_{l=1}^{L} t_l)^2}{L\sum_{l=1}^{L} t_l^2}, l \in S \tag{1}$$

In the single flow scenario, define the set $I = \{1,2,\cdots,N\}$, $i \in I$, where N the total number of packets. $BCount_i$ is the number of times that the i-th packet encounters congestion, and $delay_i$ represents the delay of the i-th packet. When the network suffers from congestion caused by burst traffic or multi-user traffic exceeding the network transmission bottleneck, $delay_i$ will increase compared with $delay_{i-1}$. The increase of $BCount_i$ will enlarge the impact of the ratio $delay_i/delay_{i-1}$, and thus tin_j can be further increased according to (2). Whenever congestion occurs, the resulting packet loss will lead to retransmission, which will crowd out the original bandwidth and lead to more serious congestion. Therefore, every time a congestion signal accumulates, the time in ProbeRTT increases exponentially. Extending the time that the congestion control algorithm stays in the ProbeRTT state will help the queue to empty. When the calculated waiting time exceeds the longest waiting time originally set by BBR, it restarts from the StartUp state. The increased time tin_j is computed as follows:

$$tin_j = tin_{j-1} * \left(\frac{delay_i}{delay_{i-1}}\right)^{BCount_i}, j = i+1 \tag{2}$$

When the congestion signal $Blocked$ is i rather than 0, it means that the amount of data is too large to be processed by the transport layer indicated by the i-th packet. Since the delay has a relatively close relationship with the network operation, the delay change signal $derate$ is used to judge the congestion situation, and its calculation is as follows:

$$derate = \sum_{k=i}^{i+3} (delay_k - delay_{k+1}) \quad (3)$$

, where $derate$ computes the sum of the delay differences between three pairs of consecutive data packets. If the result is negative, it is judged that the delay is showing an upward trend, and the network link may enter a congested state.

IBCCA will remain in the ProbeBW state when the congestion signal $BCount_i$ does not change, and when derate is a non-negative number, even if a smaller RTT value is not measured within 10 seconds. When the congestion signal $BCount_i$ becomes 1, the $derate$ is a negative number, even if the 10s detection time is not full, it enters the ProbeRTT state. In the ProbeRTT state, when the congestion signal $BCount_i$ becomes 1, and $derate$ is negative, IBCCA extends the time in the ProbeRTT state (200ms originally) or the round trip time of a data packet by *tin* milliseconds.

IV. Evaluation

To evaluate the network performance of the algorithm model in a single stream scenario, a server is used to build two virtual machines, acting as sender and receiver, respectively. The two virtual machines and their network setup are shown in Fig. 3. This experiment calibrated the time of the virtual machines and configured the environment required for QUIC transmission. In the burst traffic experiment, the experiment simulates the situation of network burst traffic by transmitting a certain amount of traffic every time interval. In the multi-stream simulation, as shown in Fig. 4, the experiment was repeated using three streams and five streams in the network link, which is compared with the single-stream scenario in Fig. 3.

In this paper, the experiment studied the performance of three congestion control algorithms including BBR, CUBIC, and the proposed IBCCA in wireless LAN. The experiment was repeated 5 times for the three congestion control algorithms, respectively, and their average network performance and behaviors to deal with congestions were recorded for further study.

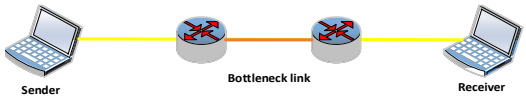


Fig. 3. Simulate network transmission scenarios for single-stream

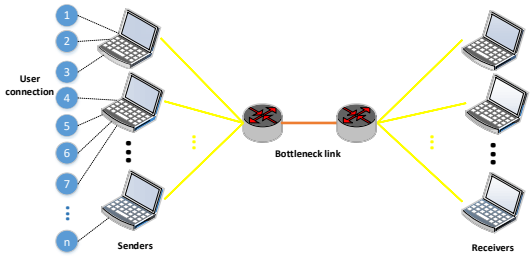


Fig. 4. Simulate network transmission scenarios for multi-stream

The simulation setup of burst flow is shown in Fig. 5. A constant sending bit rate of 90 Mbps is sent to the network by the sender. During the entire transmission, ten timestamps are selected every 10 seconds to start burst traffic with set values of duration and bit rate. This is designed to simulate burst traffic and the original data transmitted in the bandwidth-limited network. The experiment was repeated with burst traffic bit rate selected from 28Mbps, 35Mbps to 42Mbps, and duration selected from 1, 2, 3 seconds. When setting the burst bitrates to 28 Mbps, 35 Mbps to 42 Mbps, the throughputs of the three algorithms change significantly, and the performance comparison of the experimental results is also more obvious. Results are collected to study the Goodput and delay of the three algorithms under congestion. Goodput measures the throughput of original data during the transmission process. The delay is the difference between the receiving timestamp and the sending timestamp. The emergence of burst traffic may cause network congestion, competing with the original data for limited bandwidth, and thus resulting in a decrease in Goodput.

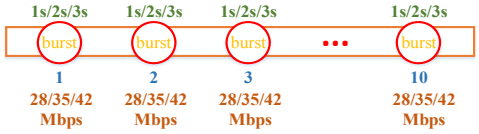
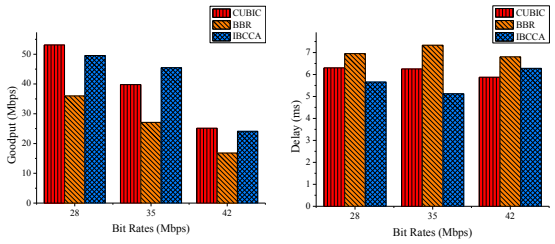
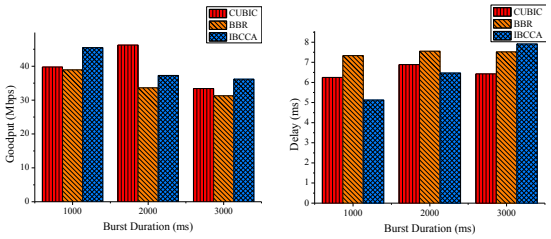


Fig. 5. Burst traffic simulation for single-stream



(a) goodput (b) delay

Fig. 6. Average goodput and network transmission delay of the three algorithms with different burst traffic of different bit rates



(a) goodput (b) delay

Fig. 7. Average goodput and network transmission delay of the three algorithms under burst traffic with different durations

In Fig. 6a, Fig. 6b, Fig. 7a, and Fig. 7b, in response to the inevitable burst traffic problem of web browsing, the experiment tested and evaluated bursts of different bit rates and burst durations. Fig. 6a and Fig. 6b keep the duration of the entire burst flow process unchanged at 1000ms and change the bit rate of the burst flows. From Fig. 6a and Fig. 6b, it can be seen that the IBCCA algorithm achieves higher network goodput, faster convergence speed, lower network round-trip time, and better network performance when dealing with burst traffic of lower bit rates. When the bit rate of burst traffic increases, the CUBIC algorithm obtains higher goodput and smaller delay

than the IBCCA algorithm. Fig. 7a and Fig. 7b maintain the average bit rate of each burst traffic at 35 Mbps and change the duration of the three burst flows to be 1000, 2000, and 3000 ms, respectively. In Fig. 7a and Fig. 7b, when the duration of network burst traffic is short, the IBCCA algorithm achieved better goodput. When the duration is longer, the goodput of CUBIC is higher. All in all, when the burst duration and average bit rate of burst traffic are within a certain range, IBCCA has better network transmission performance. According to [18], we guess that the large number of retransmissions caused by BBR's overestimation of the buffer may lead to BBR's poor performance in Goodput and fairness. A large number of retransmissions will occupy part of the bandwidth, and the bandwidth will be occupied again. It will further lead to packet loss and retransmission, resulting in lower Goodput and fairness. If the burst traffic appears in the ProbeRTT state, it is easier to synchronize when the queue is exhausted. When the burst traffic joins, the IBCCA algorithm uses the congestion block signal sent by congestion to take the state transition operation and it enters ProbeRTT state. The status can be synchronized immediately when the queue is exhausted, and the best RTT can be measured. Increasing the time in the ProbeRTT state can also reduce the sending rate to reduce packet loss and retransmission.

Table I and Table II are the average goodput of the burst traffic of different bit rates and durations, respectively. In Table I, the goodput result of the IBCCA algorithm is the highest when the burst traffic bit rate is 35Mbps. In the other two 28 Mbps and 42 Mbps cases, the goodput result of the CUBIC algorithm is the best. In Table II, the goodput result of the IBCCA algorithm is the best in the two cases of 1000ms and 3000ms, and the goodput result of the CUBIC algorithm is the best in the case of 2000ms. From Table I and Table II, the goodput of IBCCA is better than the BBR algorithm and CUBIC algorithm in the 35Mbps case. Goodput result of IBCCA is worse than the CUBIC algorithm in the situation of 35Mb data packet size and 2000ms duration.

Table III to Table VII are test evaluations of goodput, delay, and fairness for single stream and multiple stream transmissions. Table III shows that in a single stream transmission scenario, the network goodput of the IBCCA algorithm is the highest, and the network round-trip time of the BBR algorithm is the smallest. The experimental results of competition in the three streams scenario are shown in Table IV and Table V, and Table VI and Table VII contain results from the five streams scenario. In the case of three streams and five streams, the goodput of IBCCA is not significantly improved compared to that of CUBIC. However, the goodput of IBCCA achieved the best fairness for both multi-stream scenarios. When multiple streams are competing in the transmission link of the wireless LAN, the BBR algorithm will use a smaller pacing rate [7]. In this case, the detection of network bandwidth is too small, which reduces the actual goodput.

From Table III, Table V, and Table VII, it can be noted that the delay of BBR is the lowest compared to other algorithms, but its goodput is significantly lower than the other two algorithms. The delay of a single stream and multiple streams of IBCCA is numerically lower than BBR but higher than CUBIC.

It can be concluded that the proposed IBCCA algorithm has better network performance of goodput and fairness in both single stream or multiple stream scenarios. As for the delay, IBCCA does not outperform BBR, but it has a lower delay than CUBIC.

TABLE I. THE AVERAGE GOODPUT (MBPS) OF BURST TRAFFIC WITH DIFFERENT BIT RATES

Protocol	Burst Traffic Bit Rate (Mbps)		
	28	35	42
CUBIC	**53.12**	39.78	**25.14**
BBR	35.98	27.78	16.79
IBCCA	49.57	**45.49**	24.11

TABLE II. THE AVERAGE GOODPUT (MBPS) OF BURST TRAFFIC WITH DIFFERENT BURST DURATIONS

Protocol	Burst Traffic Duration (ms)		
	1000	2000	3000
CUBIC	39.78	**46.25**	33.42
BBR	38.93	33.65	31.27
IBCCA	**45.49**	37.28	**36.19**

TABLE III. THE AVERAGE GOODPUT (MBPS) AND AVERAGE NETWORK TRANSMISSION ROUND TRIP TIME (MS) OF THE THREE ALGORITHMS FOR A SINGLE STREAM

Protocols	Flow-1	
	Goodput	Delay
CUBIC	73.71	5.33
BBR	62.011	**4.29**
IBCCA	**76.49**	5.29

TABLE IV. THE AVERAGE GOODPUT (MBPS) AND FAIRNESS OF THE THREE ALGORITHMS FOR THE THREE STREAMS

Protocols	Flow-1	Flow-2	Flow-3	Fairness
	Goodput	Goodput	Goodput	
CUBIC	39.73	38.03	17.22	0.905
BBR	42.41	16.71	12.34	0.763
IBCCA	33.79	28.26	33.52	**0.994**

TABLE V. THE AVERAGE NETWORK TRANSMISSION ROUND TRIP TIME OF THE THREE ALGORITHMS FOR THE THREE STREAMS (MS)

Protocols	Flow-1	Flow-2	Flow-3	Sum
	Delay	Delay	Delay	
CUBIC	8.09	7.36	7.80	23.25
BBR	**5.90**	**7.45**	**6.24**	**19.59**
IBCCA	7.14	7.80	7.45	22.39

TABLE VI. THE AVERAGE GOODPUT (MBPS) AND FAIRNESS OF THE THREE ALGORITHMS FOR FIVE STREAMS

Protocols	Flow-1 Good-put	Flow-2 Good-put	Flow-3 Good-put	Flow-4 Good-put	Flow-5 Good-put	Fairness
CUBIC	18.46	24.90	22.63	17.08	16.80	0.975
BBR	14.95	9.53	14.91	13.77	26.51	0.888
IBCCA	16.33	17.28	20.66	24.49	20.80	**0.979**

TABLE VII. THE AVERAGE NETWORK TRANSMISSION ROUND TRIP TIME OF THE THREE ALGORITHMS FOR FIVE STREAMS (MS)

Protocols	Flow-1	Flow-2	Flow-3	Flow-4	Flow-5	Sum
	Delay	Delay	Delay	Delay	Delay	
CUBIC	9.37	8.28	9.77	9.58	12.50	49.50
BBR	**9.02**	**12.01**	**9.56**	**8.41**	**7.27**	**46.27**
IBCCA	10.40	9.50	9.28	7.59	9.92	46.69

V. CONCLUSION

This paper evaluates the network performance of CUBIC, BBR, and IBCCA algorithms under wireless LAN. Due to the smaller pacing rate, the BBR algorithm has the lowest goodput in wireless scenarios and the lowest fairness in multi-stream transmission. As the default congestion control protocol, the CUBIC algorithm has no obvious shortcomings in the network

performance of burst traffic and multi-stream competition experiments in the context of wireless LAN. But when it compares with IBCCA when the burst traffic is small or the burst duration is short, the proposed IBCCA achieves higher network goodput and lower network transmission round-trip time. What's more, in the experiment of multi-stream transmission, the IBCCA algorithm achieves higher fairness.

According to the result analysis of this paper, it is recommended that the system can switch between IBCCA and CUBIC congestion control algorithms, and actions should be made based on whether the improvement of network performance can be achieved as the burst traffic bit rate varies. However, the algorithm scheme switching may result in performance degradation due to extra complexity in protocol implementation as well as additional computation delay, protocol integration, and so on. Whether this loss can be compensated by the benefit from the algorithm switching has not been verified in this paper.

In more realistic streaming and burst traffic scenarios, the network is more complex and various other factors should also be taken into consideration as well, such as background traffic of various protocols with different congestion control schemes, multi-path routing, QoS level, priority queue, etc. Finally, the issue of fairness and friendliness between the various congestion control algorithms of TCP flow and QUIC flow is not tested and evaluated in this article. This article conducts burst traffic and multi-stream transmission experiments in wireless LAN scenarios. Researches on WLAN [20] [21] and QoS-oriented transmission [22] [23] describe scenarios such as multimedia transmission and in-vehicle network systems. In the future, we expect to conduct more experiments (such as 3D/VR video transmission with different RTT and data rates) in other scenarios (Ethernet, WAN or cellular network, etc.)

ACKNOWLEDGMENT

This work was supported by a grant from the National Natural Science Foundation of China (Grant No. 61801341). This work was also supported by the Research Project of Wuhan University of Technology Chongqing Research Institute, the Fundamental Research Funds for the Central Universities (Grant No. WUT:2021CG008), and Green Intelligent Inland Ship Innovation Programme (Grant No. MC-202002-C01-04).

REFERENCES

[1] Iyengar, Janardhan, and Martin Thomson. "QUIC: A UDP-Based Multiplexed and Secure Transport; draft-ietf-quic-transport-24," *Internet Engineering Task Force: Newark*, DE, USA. 2019.

[2] Roskind, Jim. "Quick UDP Internet Connections: Multiplexed Stream Transport over UDP." Adresse: https://docs.google.com/document/d/1RNHkx_VvKWyWg6Lr8SZ-saqsQx7rFV-ev2jRFUoVD34, December 2013

[3] Hoe, C. Janey. "Improving the Start-up Behaviour of A Congestion Control Scheme for TCP." *ACM SIGCOMM Computer Communication Review*， vol.26, pp. 270-280, October 1996.

[4] Brakmo, Lawrence S., Sean W. O'Malley, and Larry L. Peterson. "TCP Vegas: New Techniques for Congestion Detection and Avoidance." *Proceedings of the conference on Communications architectures, protocols and applications*, vol 24, pp. 24-35, October 1994.

[5] Xu, Lisong, Khaled Harfoush, and Injong Rhee. "Binary Increase Congestion Control (BIC) for Fast Long-distance Networks." *IEEE INFOCOM 2004*, vol. 4. IEEE, pp. 2514-2524, 2004.

[6] Ha, Sangtae, Injong Rhee, and Lisong Xu. "CUBIC: a New TCP-friendly High-speed TCP Variant." *ACM SIGOPS operating systems review*, vol 42, pp. 64-74, July 2008.

[7] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. "BBR: Congestion-based Congestion Control. Commun." *ACM*, vol 60, pp. 58–66, June 2017.

[8] De Cicco, Luca, Gaetano Carlucci, and Saverio Mascolo. "Understanding the Dynamic Behaviour of the Google Congestion Control for RTCWeb." *2013 20th International Packet Video Workshop*. IEEE, pp. 1-8, 2013.

[9] Narayan, Akshay, Frank Cangialosi, Deepti Raghavan, Prateesh Goyal, Srinivas Narayana, Radhika Mittal, Mohammad Alizadeh, and Hari Balakrishnan. "Restructuring Endpoint Congestion Control." *in Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pp. 30-43. 2018.

[10] Saurabh, S. Prakash, P. K. Singh, S. K. Nandi, and S. Nandi. "Is QUIC Quicker Than TCP?: A Performance Evaluation." *In Workshops of the International Conference on Advanced Information Networking and Applications*. Springer, Cham, pp. 129-138, 2019.

[11] Jan Rüth, Konrad Wolsing, Klaus Wehrle, and Oliver Hohlfeld. "Perceiving QUIC: Do Users Notice or Even Care?" *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, pp 144-150. December 2019.

[12] Minh Nguyen, Hadi Amirpour, Christian Timmerer, and Hermann Hellwagner. "Scalable High Efficiency Video Coding Based HTTP Adaptive Streaming over QUIC." *Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC*, pp.28-34, August 2020.

[13] Arisu, Sevket, Ertan Yildiz, and Ali C. Begen. "Game of Protocols: Is QUIC Ready for Prime Time Streaming?" *International Journal of Network Management*, vol 30, pp. e2063, February 2020.

[14] Kleinrock, Leonard. "Power and Deterministic Rules of Thumb for Probabilistic Problems in Computer Communications." *Proceedings of the International Conference on Communications*. vol. 43, pp. 1-43, June 1979.

[15] A. Morelli, M. Provosty, R. Fronteddu, and N. Suri. "Performance Evaluation of Transport Protocols in Tactical Network Environments." *MILCOM 2019-2019 IEEE Military Communications Conference (MILCOM)*. IEEE, pp. 30-36, 2019.

[16] Ranysha Ware, Matthew K. Mukerjee, Srinivasan Seshan, and Justine Sherry. "Modeling BBR's Interactions with Loss-based Congestion Control." *Proceedings of the Internet Measurement Conference*, pp. 137-143, October 2019.

[17] S. Zhang, W. Lei, W. Zhang, Y. Guan and H. Li. "Congestion Control and Packet Scheduling for Multipath Real Time Video Streaming." *IEEE Access*, vol. 7, pp.59758-59770, 2019.

[18] D. Scholz, B. Jaeger, L. Schwaighofer, D. Raumer, F. Geyer and G. Carle. "Towards a deeper understanding of TCP BBR congestion control." *2018 IFIP networking conference (IFIP networking) and workshops. IEEE*, pp. 1-9, 2018.

[19] R. Jain, D. Chiu, W. Hawe. "A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems, Comput." *Sci. cs.ni/9809099* (1998).

[20] Wu, G. Min, and L. T. Yang, "Performance Analysis of Hybrid Wireless Networks Under Bursty and Correlated Traffic." *in IEEE Transactions on Vehicular Technology*, vol. 62, no. 1, pp. 449-454, January 2013.

[21] G. Min, Y. Wu and A. Y. Al-Dubai, "Performance Modelling and Analysis of Cognitive Mesh Networks." *in IEEE Transactions on Communications*, vol. 60, no. 6, pp. 1474-1478, June 2012.

[22] Y. Wu, G. Min, and A. Y. Al-Dubai, "A New Analytical Model for Multi-Hop Cognitive Radio Networks." *in IEEE Transactions on Wireless Communications*, vol. 11, no. 5, pp. 1643-1648, May 2012.

[23] N. Najjari, G. Min, J. Hu, Z. Zhao, and Y. Wu, "Performance Analysis of WLANs with Heterogeneous and Bursty Multimedia Traffic." *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pp. 1-6, 2017.