# Measurement and Analysis: Does QUIC Outperform TCP?

Xiang Qin[1], Xiaochou Chen[2], Wenju Huang[1], Yi Xie[1, 2, 3*], Yixi Zhang[1]
[1]School of Informatics, Xiamen University, Xiamen, China
[2]Information & Network Services Center, Xiamen University, Xiamen, China
[3]Shenzhen Research Institute of Xiamen University, Shenzhen, China

2022 18th International Conference on Mobility, Sensing and Networking (MSN) | 978-1-6654-6457-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/MSN57253.2022.00154

***Abstract*—Many web applications adopt Transfer Control Protocol (TCP) as the underlying protocol, where congestion control (CC) plays a vital role in reliable transmission. However, some TCP mechanisms cannot cope with the requirements of new applications and ever-increasing network traffic. Therefore, people have proposed Quick UDP Internet Connection (QUIC), an excellent potential alternative based on UDP, which introduces new features to improve transmission performance and is compatible with existing CC algorithms. This paper has conducted many experiments in the testbed and actual environments to measure and compare QUIC and TCP regarding communication quality, compatibility fairness, and user experience, while considering the impacts of three typical CC algorithms: NewReno, Cubic, and BBR. QUIC outperforms TCP in most experiments for web browsing and online video, but its performance is susceptible to CC algorithms and network conditions. For example, with the Cubic algorithm, QUIC enabling the 0-RTT feature can decrease the webpages loading time by 37.11% compared with TCP. Using the BBR algorithm, both QUIC and TCP achieve high throughput, slight fluctuation, and few delayed events when playing online videos. TCP with BBR provides better fairness, while QUIC with BBR is more robust in a network with high latency or packet loss.**

***Index Terms*—QUIC, TCP, Congestion Control Algorithm, Network Measurement**

## I. Introduction

The proliferation of Internet applications and users in recent decades has led to massive network traffic. Moreover, reliable transmission and quality-of-experience (QoE) requirements have become essential Internet issues. As a reliable, connection-oriented protocol, Transfer Control Protocol (TCP) has been widely deployed in many popular applications (such as web browsing, etc.), providing error control, flow control, and congestion control. Especially congestion control plays a vital role in improving network utilization since it makes each end system control its data transmission rate and avoids data congestion in core devices like routers.

Many operating systems support TCP well. For example, the recent Linux kernel (version 5.18) can configure many congestion control algorithms (CC), including packet loss-based ones like NewReno and Cubic, delay-based ones like Nice [1], BBR (Bottleneck Bandwidth and RTT) [2], and LEDBAT (Low Extra Delay Background Transport) [3], and learning-based ones like Remy [4] and PCC (Performance-oriented Congestion Control) [5]. The learning-based CC algorithms often consume many resources and need to be standardized. So this paper focuses on the representative CC algorithms: NewReno, Cubic, and BBR, most wildly used in Linux Servers. However, TCP has some inherent defects. First, the connection establishments and TLS handshakes cost additional time before data transmission, influencing application performance, such as increasing webpages' loading time in web browsing. Second, TCP introduces queue head-of-line blocking [6], which reduces transmission efficiency and occupies many resources due to long TCP connections.

Google originally proposed Quick UDP Internet Connection (QUIC) [7], a transmission protocol based on UDP instead of TCP, and claimed that QUIC's new features could overcome TCP's shortcomings. The advantages of QUIC come from some functions, including stream multiplexing without head-of-line blocking, flexible congestion control mechanism, low-latency connection establishment (i.e., 0-RTT connections), authenticated/encrypted header and payload, etc. Moreover, Internet Engineering Task Force (IETF) has announced that the HTTP/3 standard [8] is in the "proposed standard" state, also called HTTP-over-QUIC. Then most applications worldwide can use QUIC to improve transmission performance and security, such as web servers and sensors [9].

Many researchers have studied QUIC's performance in the past few years. The papers [10]–[12] reported that QUIC's web page load time is better than TCP on Ethernet because it is not affected by head-of-line blocking. While in cellular networks, Biswal et al. [13] found that QUIC would suffer higher latency than TCP in some websites. For online videos, Google's measurements on YouTube show that QUIC reduces cache wait rates by 13.5%-18%. Kakhki et al. [14] come to a different conclusion. They measure video playback on YouTube and find that QUIC performs better than TCP over unstable networks but has no obvious advantage over stable and reliable networks. We note that the above measurements of QUIC mainly aim at Google QUIC (gQUIC) in the laboratory stage, whose results are outdated. Moreover, they neglect the role of CC, which is crucial for online video performance.

This paper investigates whether IETF QUIC [10] outperforms TCP in transmission rate, fairness, cost efficiency, and user experience. Mainly, we conduct many experiments in the testbed and actual environments to measure and analyze the performance of these two transmission protocols when applying different CC algorithms in mainstream applications:

*Yi Xie is corresponding author. E-mail: csyxie@xmu.edu.cn

978-1-6654-6457-4/22/$31.00 ©2022 IEEE
DOI 10.1109/MSN57253.2022.00154

web browsing and online video. In the testbed environment, a controlled cloud server can freely adopt TCP and QUIC while configuring any CC algorithm. Our experiments consider three representatives of the traditional, loss-based, and delay-based CC algorithms: NewReno, Cubic, and BBR, which are all compatible with these two transmission protocols. However, though QUIC and TCP adopt the same CC algorithm, they perform differently when network conditions change. Therefore, this paper considers the testbed environment with different packet loss rates and delays and the actual environment where the servers are located worldwide in various QoS networks. Moreover, this paper concerns the impacts of QUIC's new features on performance, such as the 0-RTT connection.

This paper evaluates the performance of QUIC and TCP on two popular network applications, web browsing, and video online, by conducting relevant experiments in the controllable testbed environment and the actual environment. These experiments allow the freely configuring a specified CC algorithm (NewReno, Cubic, and BBR) on the server and client sides and evaluate some application-related indicators. We specially compare QUIC and TCP from some perspectives, including the performance of web browsing and online video, the feature of fairness, and the impacts of 0-RTT comparison. Our findings include the following:

- QUIC is generally better than TCP in web browsing with the same CC algorithm. For example, fixing the Cubic algorithm, QUIC enabling the 0-RTT feature decreases the webpage loading time by 37.11% compared with TCP. In low-quality networks, QUIC combined with BBR has stronger robustness than other protocol configurations because QUIC eliminates the effects of head-of-line blocking, and BBR is not sensitive to packet loss.
- In a normal network, QUIC and TCP perform similarly in online video, and the influences of CC algorithms are not apparent. However, in a low-quality network of heavy packet loss, QUIC outperforms TCP and BBR presents a significant advantage in congestion control. For example, QUIC+BBR achieves the highest average throughput and suffers the least number of stall events. Our experiment results also show that QUIC handles the jitters of wireless networks and then outperforms TCP.
- CC algorithms have significant impacts on the fairness between QUIC and TCP. When fixing Cubic, TCP occupies more bandwidth in the early stage but quickly loses its advantage after packet loss. Then QUIC seizes the opportunity to increase its transmission rate and dominate the bandwidth. However, BBR, a feedback-driven self-adjusting speed mechanism, makes QUIC and TCP reach a balance and fairly share the bandwidth after a short period of competition.

## II. METHODOLOGY

We measure the performance of QUIC and TCP in the most widely used applications [15], [16]: web browsing and online video. First, we collect the network traffic in the testbed (with controlled servers) and the real network (with actual servers).

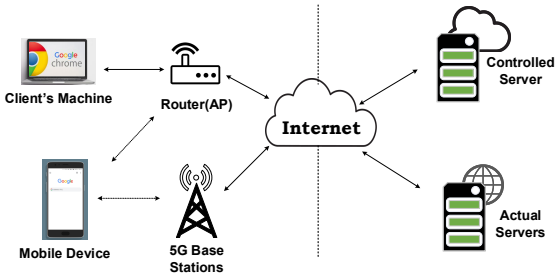


Fig. 1. Measurement environments.

Then we compute some performance indicators to analyze QUIC and TCP.

### A. Measurement Environments

We establish a platform shown in Fig. 1 to collect measured traffic data form the real environment.

The left part includes Client Machine 1 deployed on our mainland campus (Ubuntu 20.04LTS, 4GB memory, quad-core, connect to the network over Ethernet), Client Machine 2 deployed on our Malaysia campus (Ubuntu20.04, 16GB memory, quad-core, connect to the network over Ethernet), Xiaomi12 smartphone (CPU Qualcomm Snapdragon 865, Memory 8GB) connecting Internet via China Mobile 5G (Standalone). These machines browse webpages and play online videos using the extensible Chrome version. Our client side can also operate in a low-quality network like [10], [13], [14], [17], where TC or Dummynet[1] is deployed to increase 100ms additional delay and impose a 5% packet loss rate. Then we use Tcpdump and Wireshark to capture and intuitively analyze the network traffic.

The right part includes the actual website servers and a controlled clone server. It is noteworthy that many website servers in China do not support QUIC. To ensure a fair comparison, we use the HTtrack website copier[2] to clone some target webpages to the controlled server on the Aliyun ECS (Ubuntu 20.04 64bits, 2GB memory, dual-core, 10Mbps). Each cloned website has the same resources as its target website, including HTML, CSS, JavaScript, and picture files. Moreover, the controlled server supports HTTP/2 and QUIC (h3-29) via Caddy2[3] and provides security by TLS 1.3.

### B. Configurations

Two client machines operate the Chrome browser (version 105.0.5191.0), and the smartphone operates the Chrome browser (version 89.0.4324.181), those supports TCP by default. The Chrome browser can easily enable the QUIC protocol support and QUIC immediately becomes effective.

The controlled server in experiment environments supports two transmission protocols, TCP and QUIC. In the Linux kernel, we further configure three typical CC algorithms. This

[1]http://info.iet.unipi.it/~luigi/dummynet/
[2]https://www.httrack.com/
[3]Caddy2:A new kind of extensible platform.https://caddyserver.com/v2

paper generally uses NewReno, the most widely used CC algorithm, as the benchmark. And then study two advanced CC algorithms in detail: one is Cubic, the typical loss-based algorithm; the other is BBR, the typical delay-based algorithm. There are two ways to configure the CC algorithm in a server. One is to change the item "$ipv4.tcp\_congestion\_control$" in the file */etc/sysctl.conf*, the other is to use the *sysctl* command. For example, the command enables the Cubic algorithm: $sysctlnet.ipv4.tcp\_congestion\_control = cubic$.

### C. Indicators

This paper considers many performance indicators to reflect web applications' performance accurately. The page loading time and security layer handshake time reflect the web browsing performance. The online video application concentrates on the overall throughput, average throughput, number of stall events, and RTT (Round Trip Time) for performance evaluation. These indicators are defined as follows:

- The page load time (PLT) is the time it takes for a webpage to appear on the screen. The shorter PLT means a better customer experience when accessing websites.
- The overall throughput is the rate of the video file length to the period of downloading this file. Instantaneous throughput is the number of bits transmitted or received during one second when a client accesses an online video.
- The number of stall events counts the times when the player stops playing the video. A higher number means a worse user experience (QoE).

## III. Results

In the popular applications, web browsing and online video, we perform extensive measurements to analyze and compare the performance of QUIC and TCP, focusing on three typical CC algorithms, BBR, Cubic, and NewReno (regarded as a benchmark). Secondly, we study whether the 0-RTT mechanism brings advantages to QUIC.

### A. Comparison of QUIC and TCP on web browsing

This subsection uses a Chrome browser to access our controlled server, which operates 30 websites cloned from Moz's top 100[4]. This controlled server can freely select TCP and QUIC with the fine-grained control of CC algorithms. We then compare the performance of transmission protocols with 5 configurations: TCP+NewReno, TCP+Cubic, TCP+BBR, QUIC+Cubic, and QUIC+BBR, where TCP+NewReno behaves as the comparison benchmark. Fig. 2 compares the page loading time (PLT) for each website under a protocol configuration, where the vertical axis, $\beta = \frac{PLT_{benchmark} - PLT}{PLT_{benchmark}}$, records the PLT improvement rate based on the PLT under TCP+NewReno. The protocol configuration outperforms TCP+NewReno when $\beta$ is positive. The higher $\beta$, the better performance.

In a normal network environment (shown in Fig. 2 (a)), QUIC works better than the benchmark on 22 webpages

[4]https://moz.com/top500

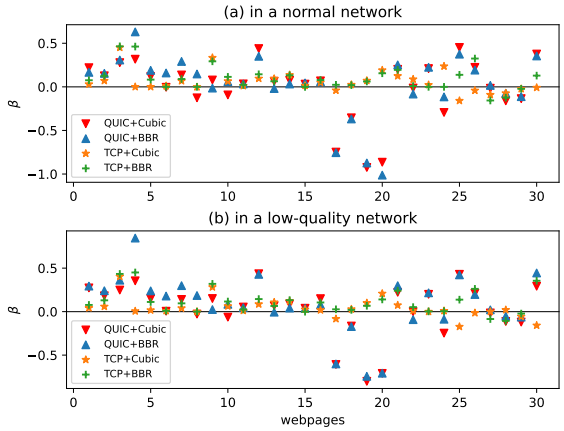


Fig. 2. Scatter plot of $\beta$, the improvement rate of PLT.

when $\beta > 0$, indicating that the overall performance of QUIC outperforms TCP. QUIC performs best on the 4th webpage (*meet.google.com*) when its $\beta$ is as high as 63.05%. When using the same CC algorithm, QUIC and TCP perform similarly. For example, fixing BBR, QUIC has a higher $\beta$ than TCP on 14 webpages. QUIC has the advantage of 12 pages, and TCP has the advantage of 12 pages. This shows that QUIC and TCP have similar performance in these 30 pages. For example, on the 12th webpage (*nytimes.com*), the PLT of QUIC+BBR ( with $\beta = 35.04\%$) is 2.94 seconds less than that of TCP+BBR ( with $\beta = 14.40\%$). We can find the same trend when fixing Cubic.

Next, we study the performance of QUIC when the network condition degrades. Many Chinese network operators restrict UDP traffic because it is more vulnerable than TCP traffic facing the stateless rapid flood attack [18]. Such a restriction unsurprisingly affects the performance and development of QUIC because it depends on UDP.

Fig. 2(b) compares the $\beta$ under different protocol configurations in a low-quality network where the router uses Dummynet to increase an additional delay of 100ms and the packet loss rate of 5%. In this low-quality network, QUIC works better than the TCP benchmark on 23 webpages when $\beta > 0$. Especially on the 4th webpage (*meet.google.com*), QUIC performs best when it's $\beta$ is as high as 84.57%. The benchmark PLTs increase because TCP suffers the Head-of-Line blocking, while QUIC successfully avoids this problem. If fixing Cubic, the $\beta$ of QUIC is higher than or equal to that of TCP on 19 webpages. For example, on the 25th webpage (*sina.com.cn*), the PLT of QUIC+Cubic ( with $\beta = 42.92\%$) is 7.35 seconds less than that of TCP+Cubic ( with $\beta = -17.28\%$). When using the CC algorithm of BBR, we find similar trends. Then QUIC wins TCP on over half of the webpages, but the improvement is obvious. Therefore, QUIC is superior to TCP against network degeneration.

Moreover, QUIC+BBR behaves best. In Fig.2 (b), QUIC+BBR performed best among the 10 pages. For example, for the 4th webpage, the PLT of QUIC+BBR is superior

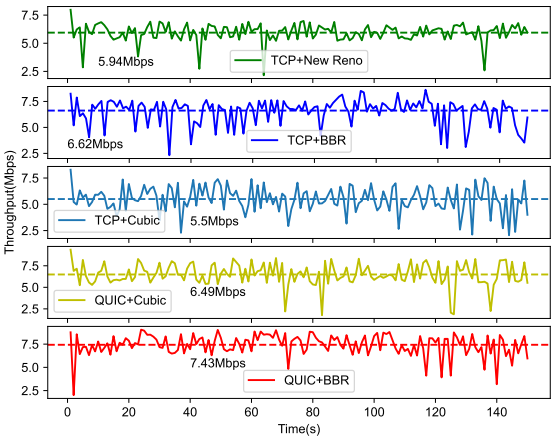


Fig. 3. Throughput of playing the 1080p video under 5 protocol configurations, with 5% packet loss.

to QUIC+Cubic by 860ms. QUIC+BBR was the worst performer on 5 of the 30 websites. But even in the worst case, QUIC+BBR is not much weaker than the next-worst configuration. For example, the PLT of the worst QUIC+BBR on the 9th page (*zh.wikipedia.org*) is only 120ms higher than that of the second worst QUIC+Cubic.

There may be two reasons for the advantage of QUIC in a network with high latency and a high packet loss rate. First, TCP easily suffers from more Head-of-line blockings when many preceding TCP segments lose while the subsequent TCP segments arrive, so the application performance is seriously affected. Whereas QUIC alleviates or even eliminates the effect of Head-of-line blocking because it quickly delivers the arrived data from the cache to the application layer. Second, most public versions of Linux default enable the SACK option, where the receiver informs the sender about the data blocks received consecutively, and then the sender speeds up the retransmission of missing data blocks. A TCP SACK can record three segment blocks at most, then slowly recover from heavy packet loss. Unfortunately, TCP's 3 ACK blocks are only good for general packet loss, and TCP can take a lot of time to deal with serious packet loss and scrambling. However, a QUIC ACK can record 256 data blocks at most [7] and then quickly recover from heavy packet loss.

### *B. Comparison of QUIC and TCP on online videos*

Firstly, we use Chrome to access the controlled server, which provides the 6'42" video with one mainstream format of MP4 1080p. This server configures NewReno (as a benchmark), Cubic, or BBR, to investigate the impacts of CC algorithms on throughput when QUIC and TCP are used. In a normal network, the throughput difference between the five protocol configurations is not apparent, so we use Dummynet to control the packet loss rate of 5%.

As shown in Fig. 3, QUIC is generally better than TCP in average throughput. For example, QUIC+BBR performs best, whose average throughput is highest at 7.43 Mbps. Moreover, BBR outperforms Cubic and NewReno in online video with the same transmission protocol. For example, TCP+BBR, TCP+NewReno, and TCP+Cubic achieve an average throughput of 6.62 Mbps, 5.5 Mbps, and 5.94 Mbps, respectively. We also note that the average throughput of TCP+BBR is slightly higher than QUIC+Cubic.

However, the throughput of TCP+BBR fluctuates while playing video. It is speculated that frequent packet loss leads to the change of bottleneck bandwidth, which affects the detection of BBR and makes the transmission rate of BBR change all the time. For example, instantaneous throughput in Fig.3 meets several sudden drops: TCP+Cubic has 7 drops, TCP+BBR has 3 drops, TCP+NewReno has 4 drops, QUIC+Cubic has 5 drops, and QUIC+BBR has 2 drops. The least number of drops explains why QUIC+BBR achieves the highest average throughput, 7.43Mbps, 14.48% higher than that of QUIC+Cubic. Therefore, QUIC+BBR performs best in a network with a high packet loss rate.

Secondly, we pay attention to the performance of online video on mobile devices. Popular Internet services have rapidly adopted QUIC in their mobile applications. Then we use Chrome on the mobile device (Xiaomi12) to play the same online video on ECS (with the bandwidth of 10Mb/s). The mobile device communication format is China Mobile 5G (Standalone) and WiFi (MERCURY AC1900), respectively. Fig. 4 records the number of stall events when the controlled server adopts different protocol configurations: TCP+BBR, TCP+Cubic, QUIC+BBR, and QUIC+Cubic. The box plot shows that when the server operates in QUIC+BBR, the mobile client suffers the least stall events. Even under the packet loss rate of 5%, QUIC+BBR performs better than other cases. The result verifies that QUIC+BBR can deal with the jitters in wireless networks.

Finally, we repeat the previous experiments using the newest video format of 4K30hz, which may replace 1080P within a few years and get much attention in recent studies. The high-definition video also stalls when encountering heavy packet loss, affecting QoE. We record the number of stall events in the 4K video playback under three network settings: no loss (i.e., a normal network), with additional packet loss rates of 5% and 10%.

Fig. 5 plots the optimization rate of stall events, $\gamma = \frac{Stall_{benchmark} - Stall}{Stall_{benchmark}}$, in one protocol configuration for online video, which uses the number of stall events under TCP+NewReno in a normal network (without imposing additional packet loss) as the benchmark. A higher value of $\gamma$ indicates fewer stall events for the protocol configuration. Without imposing additional packet loss, TCP+BBR and QUIC+BBR have the highest $\gamma$. However, under the additional 5% and 10% packet loss, the $\gamma$ of QUIC+BBR becomes much higher than the other three configurations, while TCP+Cubic has the lowest $\gamma$ (even be negative).

The $\gamma$ of QUIC+BBR and TCP+BBR for the same 4K video are about 40% under the network with an additional 5% packet loss. BBR operates well in a low-quality network because it dramatically reduces the stall events compared

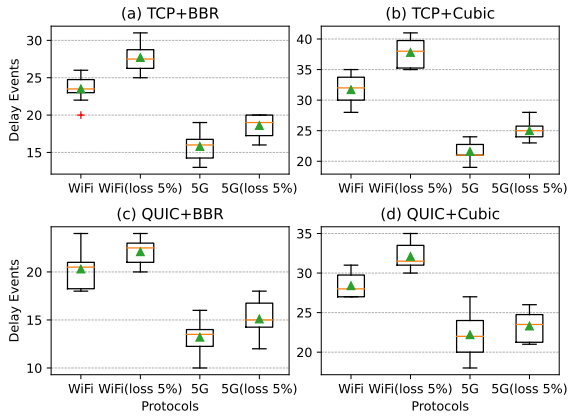


Fig. 4. The number of stall events when the mobile device plays online video under 5G and WiFi.

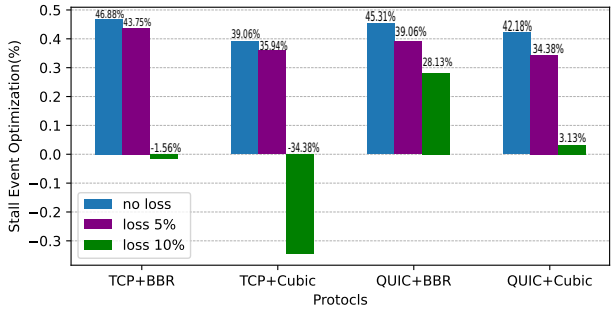


Fig. 5. Optimization rate of stall events $\gamma$ based on the TCP+NewReno benchmark, under different rates of packet loss.

with the benchmark. Under the network with an additional 10% packet loss, QUIC+BBR ($\gamma$=28.13%) still outperforms TCP+BBR ($\gamma$=-1.56%). It is easy to see that QUIC+BBR is the best choice for a low-quality network. On the other hand, NewReno, which fails under the network with an additional 10% packet loss, is inappropriate for a network with heavy packet loss.

### C. Fairness between QUIC and TCP

This subsection evaluates the fairness between QUIC and TCP, where two Chrome browsers simultaneously access our controlled server for playing the same online video. Here, Chrome 1 adopts QUIC, and Chrome 2 adopts TCP, using the same CC algorithm, Cubic or BBR. First, Cubic and BBR are typical CC algorithms on the Linux kernel. Secondly, in much literature, Cubic and BBR are used for network measurement by default. Fig. 6 plots the instantaneous throughput based on the packets captured by Wireshark in the first period of the 150 seconds. The reason to select this period is that the throughput difference afterward is not apparent and has little significance for our analysis.

**Comparison under Cubic.**

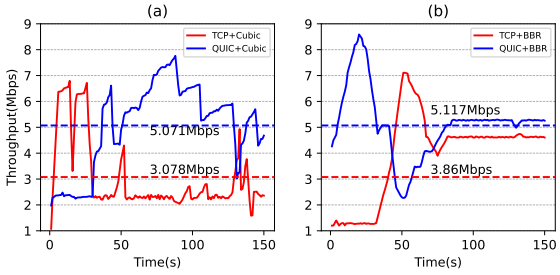


Fig. 6. Throughput of TCP and QUIC on a 10 Mbps link. (a) TCP+Cubic vs. QUIC+Cubic. (b) TCP+BBR vs. QUIC+BBR.

We compare the throughput between QUIC+Cubic and TCP+Cubic. In Fig. 6(a), the average throughput of TCP+Cubic is 3.078Mbps while that of QUIC+Cubic is 5.071Mbps (increasing 64.75%). We further observe packet loss and RTTs through Wireshark. TCP+Cubic lost 202 packets, about three times of QUIC+Cubic, which explains the throughput advantage of QUIC because Cubic is sensitive to packet loss. Moreover, the average RTT of QUIC+Cubic is about 70% of TCP+Cubic, which is consistent with the throughput improvement.

Fig. 6(a) also shows that in the early 26 seconds, TCP+Cubic occupies much bandwidth and sustains a high bandwidth utilization rate. In this period, TCP+Cubic indicates that the network's maximum bandwidth is greater than the maximum congestion window (Wmax) and enters the maximum bandwidth detection stage, where the acceleration slope on the Wmax centerline becomes high. Then we find the packet loss happens around 26 seconds, and then the cwnd of TCP+Cubic reduces to Wmax/2. At the same time, QUIC+Cubic enters the logarithmic growth and maximum detection stages, widening the congestion window and boosting the transmission volume. Therefore, the bandwidth usage of QUIC+Cubic increases, and the available bandwidth detected by TCP+Cubic decreases. After 50 seconds, TCP+Cubic decreases Wmax. As a result, QUIC and TCP become unfair because QUIC+Cubic seizes the bandwidth advantage.

**Comparison under BBR.**

Fig. 6(b) shows that the average throughput of TCP+BBR is 3.86Mbps, and the average throughput of QUIC+BBR is 5.117Mbps (increasing 32.6%) in playing the 1080p video. Instead of preempting bandwidth, BBR estimates the bandwidth-delay product (BDP) to optimize the transmission speed.

In the first 19 seconds, QUIC+BBR quickly increased the transmission rate and occupied more bandwidth than TCP+BBR. At 19 seconds, QUIC+BBR enters the buffer clearing phase [2] due to a backlog of its buffer queue. During this period, QUIC+BBR will slow down the transmission rate until the outgoing rate of the buffer is greater than the receive rate. At the 31st second, TCP+BBR detects the available bandwidth and increases its transmission rate. After 20 seconds, QUIC+BBR completes the buffer clearing phase and attempts to speed up, while TCP+BBR enters the buffer

TABLE I
COMPARISON OF PAGE LOADING TIME (PLT)

| Webpage | CC | TCP | QUIC(default) | | QUIC(0-RTT) | |
|---|---|---|---|---|---|---|
| | | | time(s) | $\alpha$ | time(s) | $\alpha$ |
| page1 | Cubic | 3.38 | 3.23 | 4.43% | 2.84 | 15.98% |
| | BBR | 3.12 | 2.98 | 4.48% | 2.64 | 15.38% |
| page2 | Cubic | 1.52 | 1.37 | 9.87% | 0.956 | **37.11%** |
| | BBR | 1.39 | 1.34 | 3.59% | 1.01 | 27.34% |
| page3 | Cubic | 4.52 | 4.34 | 3.98% | 3.87 | 14.38% |
| | BBR | 5.01 | 4.21 | **15.97%** | 3.69 | 26.35% |

clearing phase. TCP+BBR ends the buffer phase clearing at the 75th second and periodically attempts to increase the transmission rate. Finally, QUIC+BBR and TCP+BBR alternately detect bandwidth and delay and then reach balance due to the automatic feedback adjustment mechanism of BBR. Therefore, BBR achieves remarkable success in fairness for the online video application.

### *D. Does QUIC's claim of 0-RTT bring considerable benefits to performance?*

We deploy a website including three static HTML pages (Webpages objects include local server and external server) on the Aliyun ECS: Page 1 with many small-size objects, Page 2 with several large-size objects, and Page 3 with many various-size objects. First, we measure the loading time of these three pages when TCP adopts two popular CC algorithms, BBR and Cubic, respectively. Next, we repeat this measurement while replacing TCP with QUIC while enabling or disabling the 0-RTT option. Finally, we will clear the cache if QUIC disables 0-RTT (by default). Otherwise, we turn on the cache and observe the 0-RTT connections by Wireshark. Using the page loading time (PLT) of TCP as the benchmark, Table I presents the time-saving rate QUIC, $\alpha = \frac{T_{TCP}-T_{QUIC}}{T_{TCP}}$, under a specific CC algorithm.

QUIC generally outperforms TCP because all values of $\alpha$ are positive, consistent with the previous finding. Moreover, QUIC can further reduce the page loading time after enabling the 0-RTT option because the values of $\alpha$ with 0-RTT are always higher than the values of $\alpha$ with the default case. For example, the 0-RTT option shortens the page load time of Page 2 by 27.24% compared with default QUIC. We can explain it by analyzing the Wireshark traces. Here 27.6% of QUIC connections are built with 0-RTT, saving the time cost of connection establishment and TLS handshakes.

We also note a special case that TCP achieves a shorter PLT than QUIC when downloading Page 1 when TCP adopts BBR and QUIC adopts Cubic. Page 1 includes many small objects, and the client easily suffers packet loss when downloading these small objects. Cubic is sensitive to packet loss, leading to the small value of cwnd and limiting the web server's transmission rate. However, BBR's transmission rate keeps increasing because BBR is not easily affected by packet loss.

## IV. CONCLUSION

QUIC outperforms TCP in most trials due to its excellent features. With the characteristics of "eliminating head-of-line blocking" and "reducing connection time like 1-RTT or 0-RTT", QUIC has a shorter webpage loading time than TCP. QUIC also handles the jitters of wireless networks better, downloads web pages faster, and plays videos more smoothly. We also find that QUIC is more robust than TCP in low-quality networks, and CC algorithms significantly impact their transmission performance. BBR outperforms the other two CC algorithms, especially in helping QUIC and TCP reach a balance and fairly share the bandwidth.

Compared with TCP, QUIC is further involved in the application layer design. After being standardized, HTTP/3, the HTTP-over-QUIC protocol, has been supported by about 25.5% of real-world websites[5]. Therefore, we will explore QUIC's performance on novel applications using HTTP/3 while varying CC algorithms, which will help us produce the best QUIC-compatible CC algorithm.

## V. ACKNOWLEDGMENT

This work was partially supported by CERNET Innovation Project(NGII20180112), and Guangdong Basic and Applied Basic Research Foundation(No.2020A1515010709). Shaorong Fang and Tianfu Wu from Information and Network Center of Xiamen University are acknowledged for the help with the GPU computing.

## REFERENCES

[1] A.Venkataramani et al. TCP Nice: A Mechanism for Background Transfers. In *Proceedings of the USENIX OSDI*, 2002.
[2] N.Cardwell and V.Jacobson. BBR: congestion-based congestion control. *Communications of the ACM*, 60(2):58–66, 2017.
[3] D.Rossi and C.Testa. LEDBAT: the new BitTorrent congestion control protocol. In *Proceedings of the IEEE ICCCN*, 2010.
[4] K.Winstein et al. TCP Ex Machina: Computer-Generated Congestion Control. In *Proceedings of the ACM SIGCOMM*, 2013.
[5] M.Dong and T.Meng. PCC Vivace: Online-Learning Congestion Control. In *Proceedings of the USENIX NSDI*, 2018.
[6] M.Scharf et al. NXG03-5: Head-of-line Blocking in TCP and SCTP: Analysis and Measurements. In *Proceedings of the IEEE Globecom*, 2006.
[7] J.Iyengar and M.Thomson. QUIC: A UDP-based multiplexed and secure transport. RFC 9000, 2021.
[8] M.Bishop et al. HTTP/3. RFC 9114, 2022.
[9] T.Wang et al. A comprehensive trustworthy data collection approach in sensor-cloud systems. *IEEE Transactions on Big Data*, no.1, 2018.
[10] Somak R.Das. *Evaluation of QUIC on web page performance*. PhD thesis, Massachusetts Institute of Technology, 2014.
[11] A.Riddoch A.Langley et al. The QUIC transport protocol: Design and internet-scale deployment. In *Proceedings of the ACM SIGCOMM*, 2017.
[12] P.Biswal et al. Does QUIC make the web faster? In *Proceedings of the IEEE GLOBECOM*, 2016.
[13] M.Rajiullah et al. Web experience in mobile networks: Lessons from two million page visits. In *Proceedings of the ACM WWW*, 2019.
[14] A.M.Kakhki and S.Jero. Taking a long look at QUIC: an approach for rigorous evaluation of rapidly evolving transport protocols. In *Proceedings of the IEEE IMC*, 2017.
[15] M.Seufert and R.Schatz. QUICker or not?-an Empirical Analysis of QUIC vs TCP for Video Streaming QoE Provisioning. In *Proceedings of the IEEE ICIN*, 2019.
[16] N.Agarwal and M.Varvello. Mind the delay: the adverse effects of delay-based TCP on HTTP. In *Proceedings of the ACM CoNEXT*, 2020.
[17] P.K.Kharat and A.Rege. QUIC protocol performance in wireless networks. In *Proceedings of the IEEE ICCSP*, 2018.
[18] M.I.Kareem and M.N.Jasim. The Current Trends of DDoS Detection in SDN Environment. In *Proceedings of the IEEE IT-ELA*, 2021.

[5]https://w3techs.com/technologies/details/ce-http3