Network and Communication Technologies; Vol. 6, No. 1; 2021
ISSN 1927-064X E-ISSN 1927-0658
Published by Canadian Center of Science and Education

The Performance and Future of QUIC Protocol in the Modern Internet

Jianyi Wang[1]

[1] Cranbrook Schools, Bloomfield Hills, Michigan, United States

Correspondence: Jianyi Wang, Bloomfield Hills, Michigan, United States. Tel: 1-909-548-5267

Received: May 30, 2021 Accepted: July 5, 2021 Online Published: July 7, 2021

doi:10.5539/nct.v6n1p28 URL: https://doi.org/10.5539/nct.v6n1p28

Abstract

Quick UDP Internet Connections (QUIC) protocol is a potential replacement for the TCP protocol to transport HTTP encrypted traffic. It is based on UDP and offers flexibility, speed, and low latency. The performance of QUIC is related to the everyday web browsing experience. QUIC is famous for its Forward Error Correction (Luyi, Jinyi, & Xiaohua, 2012) and congestion control (Hari, Hariharan, & Srinivasan, 1999) algorithm that improves user browsing delay by reducing the time spent on loss recovery (Jörg, Ernst, & Don, 1998). This paper will compare QUIC with other protocols such as HTTP/2 over TCP, WebSocket, and TCP fast open in terms of latency reduction and loss recovery to determine the role of each protocol in the modern internet. Furthermore, this paper will propose potential further improvements to the QUIC protocol by studying other protocols.

Key Definitions:

TCP: Transmission Control Protocol. A transportation layer protocol based on the Internet Protocol and provided stability and reliability.

IP: Internet Protocol.

HTTP: Hypertext Transfer Protocol. Application layer protocol for distributing information.

SPDY: Protocol based on HTTP with improvements.

HTTP/2: Major revision of the HTTP protocol

UDP: User Datagram Protocol. A transportation layer protocol for speeding up the transfer while losing stability and reliability.

QUIC: Quick UDP Internet Connections. A transportation layer protocol based on UDP. The subject of this paper.

HTTP/3: Major revision of the HTTP protocol based on QUIC.

ISP: Internet Service Provider.

1. Introduction

1.1 What is QUIC

The QUIC protocol is a transport layer network protocol that was designed and implemented by Google in 2012 for high speed and low latency HTTP traffic. Many protocols preceded QUIC, such as SPDY and HTTP/2 via TCP, as well as improvements to the TCP protocol itself such as TCP fast open, but the fact that these predecessors were based on TCP limited their ability to manage transport layer level latency. QUIC thus came out as a replacement for TCP for handling HTTP encrypted traffic. The QUIC Protocol is based on the UDP transportation protocol. The main reason for this is that a system-level protocol implementation such as TCP would require system-wide updates on all computers using the system, and an update to all current systems would be wildly inefficient. The TCP driver is written inside the operating system, and system upgrades are usually cumulative and not as frequent as software updates. QUIC, on the other hand, is implemented in the Chrome browser based on UDP protocol, which provides flexibility and easier protocol updates.

1.2 Web Latency and Latency Reduction

Web browsing latency is one of the main factors determining the user's browsing experience (uxplanet.org, 2019). Engineers have found that latency reduction, achieved by improving the internet infrastructures such as DNS routing, and DNS redirection have reached the point of diminishing returns. Thus, engineers have shifted their focus to improving the transport protocol. There are many ways that latency can occur, such as loss of packets,

multiple handshakes, and head of line blocking delay. Loss of packets has the greatest negative impact on web accessing delay in situations where the network is congested. TCP uses a timeout to determine whether the network is congested or not. Timeouts are usually set to a large value that may affect user browsing experience by requiring more than 4 RTTs to recover from the loss. There are three popular methods to deal with loss: Proactive, Reactive, and Corrective. The proactive method redundantly transmits packets twice for connections. The reactive method retransmits the lost packet, while the corrective method sends an extra packet containing information that can be used to reconstruct the lost packet. These three methods have proved successful in dealing with loss.

1.3 The "QUIC" Way of Handling High Latency

QUIC achieved low latency and faster speed by implementing its Forward Error Correction (FEC), multiplexing connection, flow control and congestion control algorithm. FEC is an example of a corrective loss recovery mechanism. It uses a specific algorithm to generate redundant information in the sent data that allows the reconstruction of packets when loss happens without the need for retransmission. Stream multiplexing allows data to be broken apart into frames so that bandwidth can be split apart for different streams. QUIC also uses flow control to manage the buffer zones for multiple streams. It limits the size of the buffer for slow-draining streams to reserve space for other streams. These methods reduce the time spent on loss recovery in situations where the network is congested. According to UX Planet, a decrease in page load time can significantly improve viewers' browsing experience and increase user engagement. QUIC best achieves the goal of reducing latency as well as reducing the round-trip time of each request in congested scenarios. The QUIC protocol, although advanced, still has its deficiencies. There are reports of Internet Service Providers (ISPs) blocking UDP requests from the server due to potential security issues. There are also reports of QUIC utilizing excessive CPU resources on the server due to its inefficient internal structure and unoptimized cipher. Similarly, QUIC seems to be performing more poorly than TCP in situations of high bandwidth, low latency, low-loss networks due to the limitations of client-side CPUs. Improvements still can and should be made.

1.4 The Goal

This paper will propose potential improvements to the QUIC protocol that will solve some of the currently existing problems. Some of them are issues with QUIC protocol itself while others are more due to the history and nature of UDP traffic.

2. Background

2.1 Web Browsing Protocols

Before discussing QUIC, it is necessary to understand the relationship between HTTP, HTTPS, HTTP/2, SSL/TLS, SPDY, UDP, and TCP. The methods that QUIC uses were not developed all at once. QUIC Is based on the development of existing protocols, and therefore it is crucial to understand the similarities and differences between QUIC and other protocols. Currently, most web browsing protocols are based on The Hypertext Transfer Protocol (HTTP) (Fielding, Gettys, Mogul, Frystyk, & Berners-Lee,1997), which is itself based on the Transmission Control Protocol (TCP) (Postel, 1981). HTTP3 over QUIC, on the other hand, is based on the User Datagram Protocol (UDP) (Langley et al., 2017). Although these approaches are vastly different, the goal and technology used are of the same origin.

2.2 TCP

TCP is one of the main transport protocols that complements the Internet Protocol (IP). It offers reliability, stability, and congestion control based on losses. Although TCP does provide this congestion control, the fact that loss occurs before any action can be taken to prevent that loss makes TCP somewhat inefficient in terms of loss recovery. The HTTP protocol is an application-level protocol that allows users to easily access hypertext documents and other resources on a web page. The HTTP protocol is implemented above the TCP protocol. It is widely deployed and acts as the basis of other related protocols such as WebSocket, HTTPS, and newer versions of HTTP (HTTP2/3). The problem with HTTP is that security and maximum utilization of bandwidth were not well considered in its creation since it was created in the earliest stage of the Internet.

2.3 SSL

Secure Sockets Layer (SSL) (Freier, Karlton, & Kocher, 2011) and Transport Layer Security (TLS) (Rescorla & Dierks, 2018) are security protocols designed to establish authenticated and encrypted connections between computers. Hypertext Transfer Protocol Secure (HTTPS) (Rescorla & Schiffman, 1999) is the secure version of HTTP that uses the security protocol TLS to encrypt HTTP traffic and provide security between the client and the server. However, HTTPS requires a full round-trip time delay in exchanging the key pairs between the client and

the browser.

2.4 HTTP/2

HTTP/2.0 or HTTP/2 (Belshe, Peon, & Thomson, 2015) derives from early experiments by Google known as the SPDY protocol. HTTP/2 is highly compatible with the original HTTP/1.1 protocol. At the same time, it decreases latency and improves page load speed with stream multiplexing, server push, and header compression, and solves the head-of-line blocking issue of HTTP. It is also highly compatible with the HTTPS protocol in that it requires fewer TLS handshakes. The main difference between HTTP/2 and QUIC is that HTTP/2 works on TCP and relies on TCP for congestion control and loss recovery. Since TCP is built into the system kernel, it generally takes a lot of effort from the developers for an update to get deployed on all systems. Because of this, the development of HTTP is limited by the development of TCP.

2.5 TCP Fast Open

Even though an update to TCP takes a long time to be deployed, there has been an update pushed to TCP that attempts to increase its performance. TCP Fast Open (TPO) (Cheng, Chu, Radhakrishnan, & Jain, 2014) is an extension of the TCP protocol. Differing from the abovementioned protocols, TPO sends a TFO cookie along with its initial SYN request to the server so that the server can authenticate the client and start sending information before the final ack request. In this way, a full round-trip time delay is reduced. QUIC has a session system like that of TPO that saves the initial handshake RTT.

3. Comparison

3.1 Handshake Latency

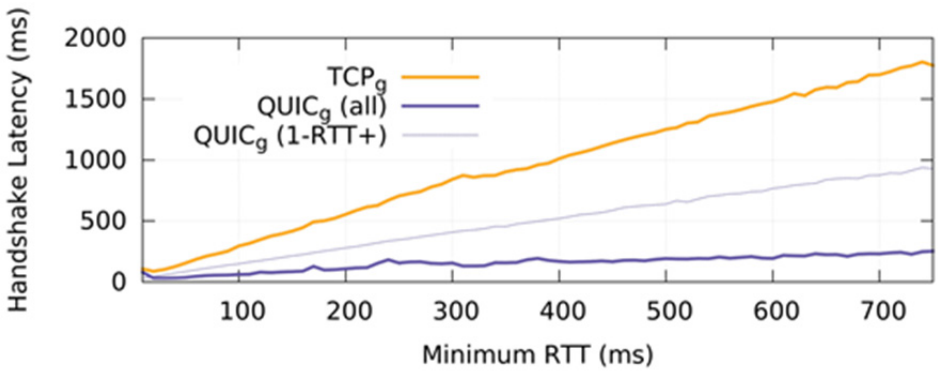

Figure 1. Handshake Latency vs Minimum RTT

The most important aspect of QUIC is its handshake latency. QUIC can establish most secure connections within 1 RTT, which appears to be almost entirely unaffected by increased minimum round-trip time. QUIC can save a huge amount of time upon connection establishment and could provide stability in situations where the latency is high.

According to this research by Google (Figure 1), the performance of QUIC under realistic scenarios appears to be better than TCP/TLS. The difference in terms of latency between QUIC and TCP increases as the minimum RTT increases. According to Figure 2, QUIC users with high latency networks can achieve the same search latency, video rebuffer rate and video latency as TCP users with much lower latency.

3.2 Video Latency

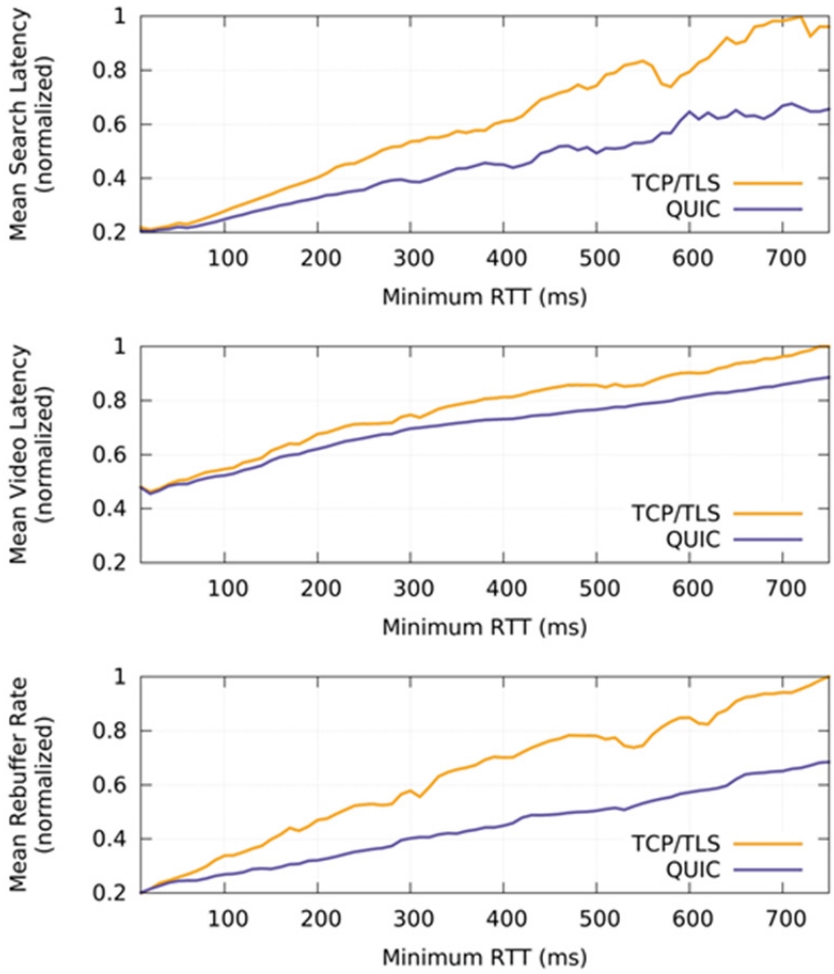

Figure 2. Mean Rebuffer Rate, Video Latency and Search Latency vs Minimum RTT

The abovementioned results are collected from YouTube video playbacks by Google. Although the data is authentic, its scope is too narrow, and it does not contain all the necessary information to determine the overall performance of QUIC. For instance, the data doesn't compare the total time consumption of large file transfers under different minimum RTT scenarios. Because of this, I am suggesting a potential method of comparing the performance of HTTP3 over QUIC to HTTP2 over TCP that could more accurately show the benefits and deficiencies of QUIC. First, the dataset will be created using the Linux dd command. The command will generate 1 kb, 1 mb, 10 mb, 100 mb, and 1 gb sized files containing random data. The source server will run Nginx web server with HTTP2 over TCP and HTTP3 over QUIC separately. After the server turns on, the files will be requested by client servers located in two different areas, one located near the source server and the other located far away from the source server, and the response time and transfer rate will be recorded.

4. Improvements

4.1 Improvements to QUIC

QUIC is almost perfect in terms of speed and latency, but improvements can be made to make QUIC an even better protocol. First, the difference in performance between QUIC and TCP becomes pronounced only when the minimum RTT is high. But in most cases in most parts of the world, RTTs are exceptionally low when the user is connected to a nearby server. QUIC usually wastes copious CPU resources on it.

4.2 Limitations of UDP

UDP traffic is widely known for its use in Dos (Denial of Service) attacks. Due to the tremendous impact of the attacks on internet infrastructure, most datacenters limit or even completely block UDP traffic from being sent &

received. As we talked about in this paper, QUIC is based on UDP and can be similarly affected by those firewall rules deployed by datacenters.

Figure 3. Famous Cloud Service Provider Limiting QUIC Traffic (https://quic.nginx.org/quic.html)

Figure 3 shows how a famous cloud service provider blocks quic traffic after detecting a large number of requests and traffic. This is not the case for just browsing a web page, but in scenarios when downloading large files or streaming long videos. In this case, the bandwidth is fully occupied and causes the firewall to detect and block the traffic.

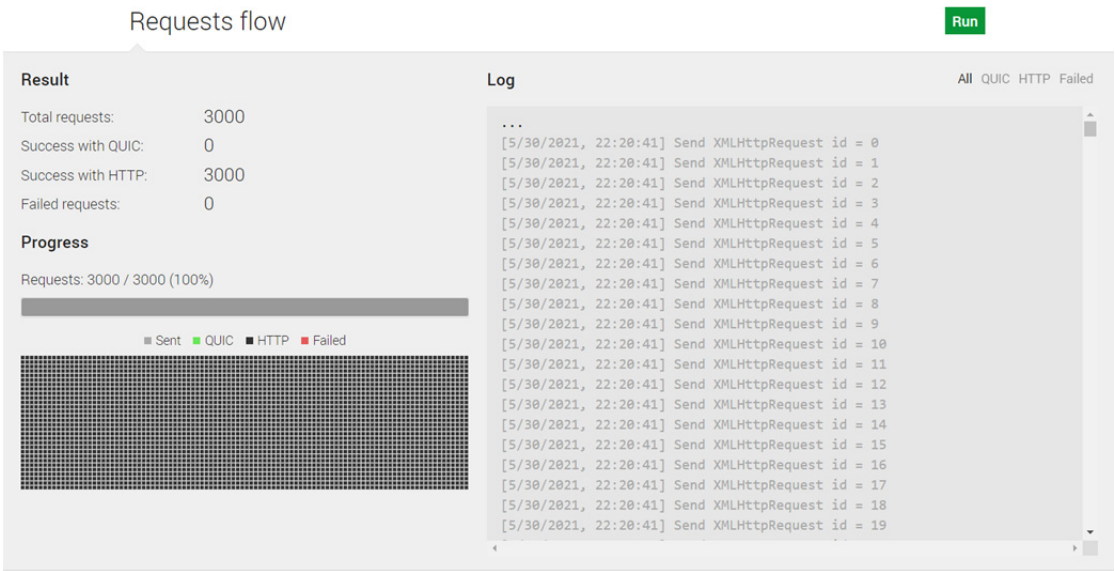

Figure 4. School Network Completely Blocking QUIC Traffic (https://quic.nginx.org/quic.html)

Figure 4 shows how the networking system in my school is completely blocking all QUIC traffic (essentially UDP traffic). It is quite common for big organizations to completely block udp traffic within their local network to protect the network against malicious attacks.

4.3 Potential Solution

Datacenters and Internet Service Providers should allow QUIC traffic to bypass the firewall by whitelisting QUIC headers and identify them as QUIC traffic instead of traditional UDP traffic. Also, QUIC should be able to prevent

malicious traffic from pretending to be normal QUIC traffic. By reducing restrictions on QUIC traffic, the availability and deployability of QUIC will drastically increase. And quic will be available to much more users and useful in more scenarios.

Acknowledgments

I would like to thank my mother for her continuing support for my research project.

References

Langley, A., Riddoch, A., Wilk, A., Vicente, A., Krasic, C., Zhang, D., ... & Roskind, J. Kulik, Joanna, Patrik Westin, Raman Tenneti, Robbie Shade, Ryan Hamilton, Victor Vasiliev, Wan-Teh Chang, and Zhongyi Shi. 2017. The QUIC transport protocol: Design and Internet-scale deployment. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication. ACM* (pp. 183-196). https://doi.org/10.1145/3098822.3098842

Belshe, M., Peon, R., & Thomson, M. (2015). *Hypertext transfer protocol version 2* (HTTP/2). https://doi.org/10.17487/RFC7540

Cheng, Y., Chu, J., Radhakrishnan, S., & Jain, A. (2014). *"TCP Fast Open", RFC 7413*. https://doi.org/10.17487/rfc7413

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., & Berners-Lee, T. (1999). *Hypertext transfer protocol–HTTP/1.1*. https://doi.org/10.17487/rfc2068

Freier, A., Karlton, P., & Kocher, P. (2011). *The secure sockets layer (SSL) protocol version 3.0* (Vol. 11). RFC 6101. https://doi.org/10.17487/rfc6101

Balakrishnan, H., Rahul, H. S., & Seshan, S. (1999, August). An integrated congestion management architecture for Internet hosts. In *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication* (pp. 175-187). https://doi.org/10.1145/316194.316220

Nonnenmacher, J., Biersack, E., & Towsley, D. (1997, October). Parity-based loss recovery for reliable multicast transmission. In *Proceedings of the ACM SIGCOMM'97 conference on Applications, technologies, architectures, and protocols for computer communication* (pp. 349-361). https://doi.org/10.1109/90.720869

Postel, J. (1981). *Rfc0793: Transmission control protocol*. https://doi.org/10.17487/RFC0793

Rescorla, E., & Dierks, T. (2018). *The transport layer security (TLS) protocol version 1.3*. https://doi.org/10.17487/RFC8446

Rescorla, E., & Schiffman, A. (1999). The secure hypertext transfer protocol. *IETF Request for Comments, RFC, 2660*. https://doi.org/10.17487/rfc2660

Luyi, S., Jinyi, F., & Xiaohua, Y. (2012, August). Forward error correction. In *2012 Fourth International Conference on Computational and Information Sciences* (pp. 37-40). IEEE. https://doi.org/10.1109/ICCIS.2012.158

uxplanet.org. (2019, December 29). *How page speed affects Web User Experience. UX Planet*. Retrieved February 27, 2021, from https://uxplanet.org/how-page-speed-affects-web-user-experience-83b6d6b1d7d7