QUIC Protocol Performance in Wireless Networks

Prashant K Kharat, Aniket Rege, Aneesh Goel, Muralidhar Kulkarni

Abstract— **Google's Quick UDP Internet Connections (QUIC) transport layer protocol was developed in 2013 as a successor to its own SPDY networking protocol, which itself led to the formation of the HTTP/2 standard. QUIC's main motives were to take the advantages of TCP/IP and HTTP/2, and build them over UDP, in terms of reliability, flow control, and congestion control. The primary objective of this paper is to explore QUIC functionalities to suggest techniques to improve throughput, speedup and efficiency in wireless networks. The experimental results were established on a local test bed setup connected to a wireless access point in a campus network environment. Experimental results show that QUIC performance in the form of throughput and speedup over TCP/IP in live network environment. The fairness of QUIC in competing flow situations is also examined, and found to perform well in long life traffic. We also propose the reintroduction of FEC for minimization of retransmission latencies.**

***Index Terms* - Congestion control, Forward error correction, QUIC, Wireless networks.**

I. INTRODUCTION

In today's fast paced world, the rate and quality of information transfer has ascended to paramount importance. By 2021, Cisco Visual Networking Index estimates that annual global IP traffic will hit 3.3 ZB, of which 63% will be in the form of wireless and mobile traffic. Furthermore, the report states that Internet Protocol (IP) video traffic will make up 83% of all consumer IP traffic, of which 13% will be live streaming video. As mobile and IP enabled devices grow into an Internet of Things (IoT) driven by 5G technology, IP traffic is bound to increase even faster in the coming years.

In this scenario, it becomes ever more relevant to establish networking protocols to handle the rapid growth in IP traffic usage. The popular Hypertext Transfer Protocol (HTTP), published in RFC 2616 in 1999, had become outdated with the changes in Internet usage and traffic. To this end, Google developed its SPDY web transfer protocol [4] using multiplexed TCP streams and header compressions on top of HTTP 1.1 with the aim of significantly improving Page Load Time (PLT). In 2015 the Internet Engineering Task Force (IETF) released the largely SPDY based HTTP/2 protocol in RFC 7540 [5] as a successor to the HTTP 1.1 protocol. HTTP/2 are application layer protocols running on top of Transmission Control Protocol (TCP) in the transport layer, which has its own limitations. Its connection oriented service with three way handshake increases the Round Trip Time (RTT) for an HTTP request, which degrades twofold if SSL/TLS encryption is also present. An alternative to TCP, the connectionless User Datagram Protocol (UDP), has not gained much popularity due to its lack of reliability and Quality of Service guarantees. Google in 2013 proposed Quick UDP Internet Connections (QUIC) Protocol [6] over UDP in the transport layer, as an alternative to HTTP/2 over TCP/IP. This protocol is relatively unexplored, especially in the wireless connection domain.

The primary aim of this paper is to explore the network characteristics of QUIC in a live wireless network compared and propose a scheme for using Forward Error Correction (FEC) to improve retransmission latencies, congestion, and throughput of the connection. Present study also examine the fairness of QUIC protocol with respect to TCP/IP in a live network scenario for further insight into the above.

This paper is organized as follows. Section II describes the background of HTTP/2 over TCP/IP[1] and QUIC over UDP. Section III details the measurement environment used to compare QUIC's performance with TCP based protocols and deduce the results presented in Section IV. Finally, we summarize and conclude our work in Section V.

II. Fundamentals of QUIC

The fundamental workings of TCP/IP and UDP transport layer protocols are essential in acquiring a deeper understanding into the defining features of QUIC and HTTP/2, the major protocols deployed in servers across the World Wide Web. QUIC and HTTP/2, the former deployed primarily on Google's servers, build on top of these protocols to achieve higher security, efficiency, and reliability. The key underlying features that enable these protocols to efficiently transport data over communication channels is explored in brief in this section.

A. HTTP/2 over TCP/IP

1) TCP/IP Fundamentals: As seen in [1]

- Connection-Oriented Service: Before any communication can takes place between a client and server, a connection must be established between them. Each TCP connection is identified by a 4-tuple: source TCP port number, source IP address, destination TCP port

Prashant K Kharat is with Department of Electronics and Communication Engineering, National Institute of Technology, Karnataka, India. (e-mail: prashant.kharat@walchandsangli.ac.in)

Aniket Rege is with Department of Electronics and Communication Engineering, National Institute of Technology, Karnataka, India. (e-mail: aniketrg7@gmail.com)

Aneesh Goel is with Department of Electronics and Communication Engineering, National Institute of Technology, Karnataka, India. (e-mail: goelaneesh29@gmail.com)

Muralidhar Kulkarni is Professor in Department of Electronics and Communication Engineering, National Institute of Technology, Karnataka, India. (e-mail: mkulkarni@nitk.ac.in)

[1]All further mentions of HTTP/2 should be read as HTTP/2 over TCP/IP. A comparison of QUIC and HTTP/2 is thus equivalent to QUIC vs TCP/IP

978-1-5386-3521-6/18/$31.00 ©2018 IEEE

number, and destination IP address. The connection is terminated upon completion of the communication session.

- Streaming Service: Once a TCP connection is established between two application processes, the sender will write a stream of bytes into the connection and the receiver will read these bytes out of the connection.
- Reliable Service: TCP guarantees delivery of every single byte in order, without any duplication. In order to achieve this, TCP uses the acknowledgment (ACK) mechanism to check if the transmitted data has been received correctly by the receiver. Data unacknowledged within a threshold period is retransmitted later.
- Flow Control and Sliding Window: this mechanism prevents a fast sender from overloading a relatively slower receiver. Each TCP connection has a buffer to store data temporarily, which is cleared as soon as data is successfully sent to the corresponding receiver. The buffer may overflow if it cannot be emptied at a rate equal to incoming data from sender. In this case, TCP employs flow control to limit incoming data rate and prevent buffer overflow.
- Congestion Control: mechanisms applied to prevent buffer overflow in intermediate routers. The fundamental principle is to adjust the transmission window at the sender such that buffer overflow is not encountered at any intermediate routing point in the connection, nor at the endpoints.

2) UDP Fundamentals:

- Connectionless Service: UDP does not implement connection establishment and connection termination. UDP initiates the first-time connection between the sender and receiver using a cryptographic token, and provides a mechanism for the application to send short messages to the destination in a stream mechanism.
- Datagram-Oriented Service: As a datagram-oriented service, UDP cannot accept a stream of data from the application and segment them for transmission. The application is required to supply data to UDP for transportation as an independent datagram.
- Unreliable Service: UDP suffers from a lack of reliability due to no acknowledgment or retransmission mechanisms, and no sequence numbers to identify each datagram. Therefore, a UDP sender will not know if a datagram was lost on the way. There is no flow control either, meaning that a UDP receiver may experience buffer overflow without responsive transmission window adjustment. This is a major drawback of native UDP protocol.

B. QUIC Protocol

QUIC was proposed in an effort to provide multiplexing and flow control equivalent to HTTP/2, security equivalent to TLS, and connection semantics, reliability, and congestion control equivalent to TCP/IP [8].

As seen in Fig. 1, a connection is initiated for the first time by the client sending a non-encrypted Client Hello

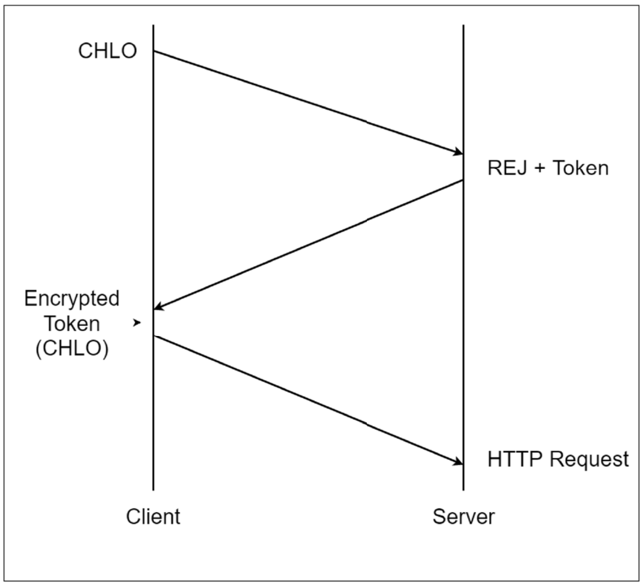

Fig. 1: Connection setup procedure of QUIC protocol

(CHLO) packet to the server. The server responds by creating a cryptographic token which is based on the connection ID and sends it back to the receiver in a reject (REJ) packet. This token goes in the cache of the client for fast access and is used to encrypt all HTTP requests that the client sends to the server. The same token is used for sending any successive HTTP requests to the server, thereby saving connection setup time. Reusing the key or token minimizes the need for a handshake to a given server.

As seen in [7], The Round Trip Time (RTT) is a major drawback in TCP/IP which can be overcome by using UDP as it is connectionless. TCP needs one round-trip to open the connection before the actual web request can be made with HTTP. It would need at least an extra round-trip for establishing transport layer security (TLS). Further, if this is the first interaction between the client and server TCP would need an extra authentication phase thereby adding one more round-trip making it three in total.

On the other hand, with QUIC a client can achieve 0-RTT connection cost when connecting to a server if a connection has already been established between the two. This is achieved by using a connection ID in every packet instead of the conventional four-tuple IP address used in TCP/IP. The encryption mechanism used in QUIC aims to change the standard TLS method which increases one RTT. The new design resembles Datagram Transport Layer Security (DTLS).

C. Congestion Control

Congestion control consists of mechanisms applied to prevent buffer overflow in intermediate routers. The fundamental principle is to adjust the transmission window at the sender such that buffer overflow is not encountered at any intermediate routing point in the connection.

Additive Increase Multiplicative Decrease (AIMD) is the primary algorithm used in TCP for governing the sending rate where the sender increases the window size in a linear additive fashion upon receiving each acknowledgment and decreases the size exponentially whenever there is a packet loss. As seen in [1], there are two parts in the algorithm:

- Congestion Avoidance (CA): There is an assumption that the Advertised window (receiver) is much larger than the Congestion window (*cwnd*). Initially in the slow start phase, the window size practically gets doubled after each RTT, allowing twice as many segments to be transmitted. This exponential increase may lead to congestion if not checked. To avoid this, TCP implements Congestion Avoidance algorithm, which forces a linear Additive Increase (AI) of the congestion window after it reaches a threshold (ssthresh) which can be adjusted dynamically. Generally, the congestion window is incremented by 1/*cwnd* each time an acknowledgment (ACK) is received. This way the window is effectively increased by one every RTT.
- Multiplicative Decrease (MD): This controls the dynamic variable ssthresh mentioned above. TCP sets ssthresh to half the current *cwnd* each time a timeout occurs (at timeout *cwnd* itself is set to one segment to force a slow start) down to a minimum of ten segments (starting point for AI step). Therefore, if there are consecutive timeouts (severe network congestion), Multiplicative decrease reduces the sending rate exponentially.
- In QUIC protocol CUBIC [3] approach is used to control congestion.

D. *FEC and Retransmissions*

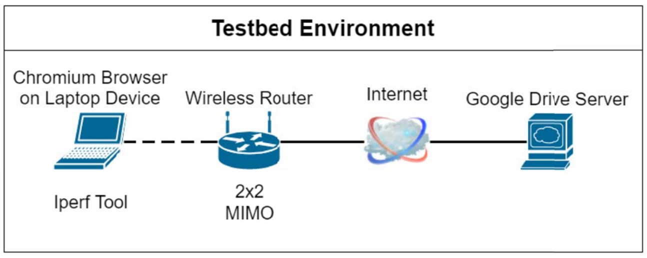

Fig. 2: Testbed for performance evaluation

An important feature of QUIC protocol is its provision of Forward Error Correction (FEC) to minimize retransmission latencies. FEC enables recovery of lost packets and can improve latency at the cost of increasing bandwidth. It uses an exclusive-or (XOR) technique to recover lost packets without having to wait for retransmission of the packets (which is performed directly in TCP/IP). Packets are labeled in groups by an additional XOR packet with redundant bytes which can be used together with the rest of the group packets to recover any single lost packet. This mechanism is only useful for recovering one lost packet/group. If multiple packets are lost QUIC will have to revert to retransmitting lost packets. If no packets are lost, then this additional XOR packet will be an unnecessary overhead and thus ineffective.

TABLE I: Parameter space used for experimentation

Category	Parameter	Value
Network Parameters	Bandwidth	10 Mbps, 18 Mbps, 6 Mbps
	Link Capacity	72 Mbps/client for 2.4 Ghz band 433 Mbps/client for 5 Ghz band
	RTT	62 ms, 54 ms, 48 ms
	Packet Loss	0%
Server Side	Video File Size	100 KB, 1 MB, 10 MB
	No. of Flows	2 QUIC, 2 HTTP/2

Therefore, it would be useful to know the distribution of lost packets so as to adapt FEC functionality accordingly, such as through channel modeling. Currently QUIC's implementation of FEC is seen as a drawback, as current tests have shown either no significant improvement, or worse performance for throughput on enabling FEC[2][11], [16].

E. *Fairness*

To improve performance in terms of utilization of network resources and minimization of network congestion, congestion control algorithms such as CUBIC [3] were developed. But the effectiveness of these algorithms is not only based on how effectively they handle congestion and maximize resource utilization, but also fairness of resource usage (everyone gets a fair share) and efficiency (network resources are used well) of congestion control algorithms. To achieve both these objectives, D. Chiu, R. Jain [2] formulated a fairness measure (F) known as Jain's Fairness Index, which is a single dimensionless fraction [1]. This index is a network resource consumption function by each user sharing the same path, and is shown below:

$$F(x_1,x_2,\ldots,x_n) = \frac{(\Sigma_{i=1}^{n} x_i)^2}{n * \Sigma_{i=1}^{n}(x_i)^2} \qquad (1)$$

$F \rightarrow$ Fairness Index function

$x_i \rightarrow$ throughput for the ith connection

$n \rightarrow$ number of flows

Jain's Fairness Index is a predominant fairness measure for TCP flows. This fairness index ranges from 1/n to 1. The value of this Index tends to 1 if and only if each flow has an equal share, and tends to 1/n if a single flow acquires all network resources. Generally AIMD Algorithm converges to good fairness and efficiency values.

III. Measurement Environment and Testbed

During experimentation, main objectives were to examine efficiency improvement of QUIC in terms of throughput in low BDP networks and fairness in a wireless environment. The details of the measurement environment used to test the performance of QUIC in varied scenarios is detailed in this section and sketched in Fig. 2. The tests were performed on a live wireless network, with router details also noted[3].

[2]Compare to benefits implementation code level complexity is more, consequently FEC funcationality is removed form QUIC in early 2016.

[3]Tests were run on a 2:1:1 load balanced network supplying 3 lines of 1 Gbps, 450 Mbps, 450 Mbps capacity to the entire university campus

A MacBook Pro laptop device running chromium browser built from source was used to carry out measurements[4].

On the server side, Google server (drive) has been used to serve requests from the client due to the relative rarity of live QUIC enabled web servers. Google Services, such as Drive and YouTube, use QUIC as a native protocol, as opposed to other web servers where QUIC has limited deployment. It is also possible to build a QUIC server-client topology from the chromium code base as shown in [9]. This implementation of a QUIC server has been used to test results from the laptop device specified above. This has been noted that implementation was meant for integration testing, and not to test performance at scale. However the results of G. Carlucci, L. De Cicco, and S. Mascolo [11] suggest this may be a significantly relevant method to measure QUIC performance, as validated by Srivastava A. [12].

Previous literature, such as Megyesi, P., Krmer, Z. and Molnr, S. [7] and S. Cook, B. Mathieu, P. Truong, and I. Hamchaoui in [13], has analyzed the performance of QUIC with respect to TCP based HTTP/1.1, SPDY, and HTTP/2 in varied real-world scenarios, including the presence of packet losses and network congestion. These studies have primarily taken place for wired connections, and strive to compare these to live wireless network tests, as well as test fairness of QUIC in these scenarios.

The procedure was carried out to test competing flows serviced by both QUIC and HTTP/2. QUIC flows were created using both the QUIC client built from chromium source, as well as Google Chrome browser. Whereas TCP flows were created with Opera and Mozilla Firefox. Separate browsers were required to create separate flows as the multiplexing nature of HTTP/2 causes the use of a single TCP/IP connection, which limits our ability to test durability under multiple flows. Video files of 100 KB, 1 MB and 10 MB size were uploaded to Google Drive to be serviced by any and all flows. This experimental setup cannot accurately test performance of a live Google server servicing thousands and thousands of flows simultaneously, but hypothesize that results obtained from our experimentation should scale reasonably well for servers handling considerably more flows.

The available bandwidth and ping (RTT) were tested during each epoch with Speedtest network monitoring tool, and each correspond to the video file below directly (For example: 18 Mbps bandwidth and 54 ms RTT for the 1 MB file download, and so on) [10]. The parameter space of the experiment is detailed in Table I.

IV. Experimental Results

The tests were run over a total of five epochs to minimize variance outside the scope of a direct throughput comparison of QUIC and TCP/IP. The fairness of utilization of available bandwidth by the four different flows (2 TCP and 2 QUIC) was calculated according to Jain's Fairness Index given in equation (1).

[4]Intel Core i5 CPU @2.5 GHz with 4 GB RAM running MacOS Sierra 10.12.6 (64 bit). Chromium Version was 63.0.3229.0 (64 bit), Darwin kernel version was 16.7.0

TABLE II: Throughput and Speedup for QUIC vs TCP/HTTP2 for various video file sizes

No. of Flows	Parameters	100 KB	1 MB	10 MB
1	QUIC Throughput (Mbps)	0.82	0.88	0.68
	TCP Throughput (Mbps)	0.43	0.73	0.45
	Speedup (QUIC over TCP)	1.90	1.21	1.51
4	QUIC Throughput (Mbps)	0.39	0.56	0.83
	TCP Throughput (Mbps)	0.77	1.31	1.16
	Speedup (QUIC over TCP)	0.51	0.43	0.72

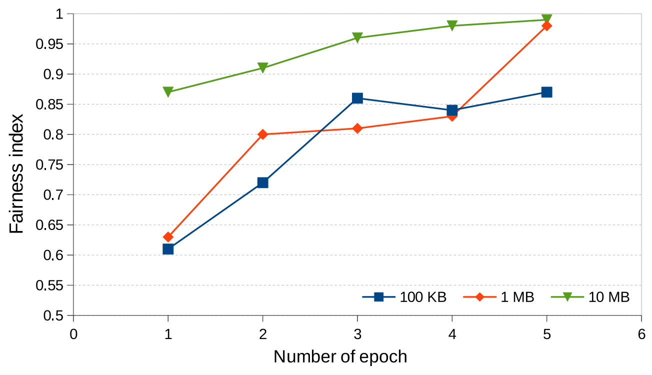

Fig. 3: Jain's Fairness Index over all 5 epochs

The results are summarized in the form of throughput characteristics for QUIC over UDP and HTTP/2 in Table II. As is evident, the throughput characteristics are drastically different when QUIC and TCP have competing flows, as opposed to none at all. With extensive experimentation in live network results shows that QUIC throughput outperforms TCP/IP, especially for single dominant QUIC flow. In case of multiple streams (flows), HTTP/2 creates multiple dedicated connections to serve each flow. However QUIC uses multiplexed UDP streams in addition to dedicated connections similar to its predecessor SPDY, which multiplexes TCP streams [4]. This causes a fall in speed measured in terms of throughput improvement of QUIC with respect to TCP. The throughput and speedup of QUIC over TCP/HTTP2 comparison in the case of competing flows as well as single flows is also tabulated in Table II.

The behavior of fairness during competing flows is shown

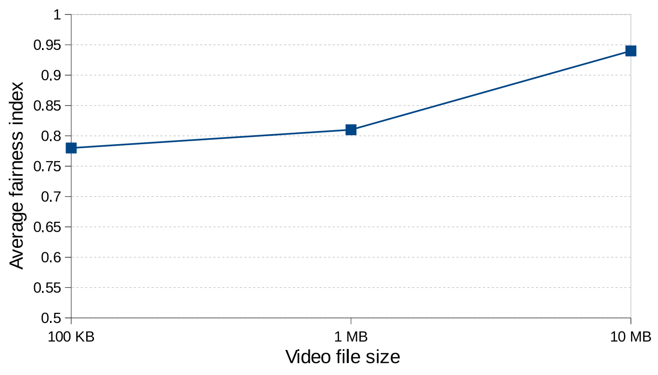

Fig. 4: Average Fairness Index over all video file sizes

in Fig. 3. The average behavior of fairness is also shown in Fig. 4, where a clear increase in fairness with increase in file size (long live traffic) can be observed. The requests serviced in Opera and Firefox via TCP/IP reserved a larger portion of the bandwidth for the smaller files, and were the primary contributor to the relatively larger value of "unfairness" in the smaller files.

However, in case of multiple competing flows, TCP/IP achieves higher throughput than QUIC. The experimental results indicate that a dominant QUIC flow improves throughput and overall speedup of a wireless connection. QUIC shows advantages over SPDY (the building block for HTTP/2) as it does not suffer from SPDY's Head of Line (HOL) blocking, as verified for a wired connection in [7]. This is due to UDP not requiring in-order packet delivery. Fairness results seen in Fig. 3 and Fig. 4 show that QUIC becomes fairer for elephant traffic (long live traffic, 10 MB) than mice traffic (short live traffic, 100 KB), as it requires some convergence time.

As this experimentation was carried out in a wireless environment, packet loss becomes an incredibly significant measure of importance for quality of a connection as opposed to over a wired link. The FEC feature of QUIC would aid in minimizing local loss due to bit error, one of the very reasons it was created. However due to inconclusive results as to its efficiency [11], the FEC functionality of QUIC was removed from Chromium by Google in 2016. This experimental results demand the re-addition of a modified FEC functionality to QUIC protocol to improve upon retransmission latency.

QUIC employs TCP CUBIC and pacing approach to avoid congestion and retransmission latencies [3]. Congestion at intermediate routers is difficult to improve in a TCP/IP implementation, as it requires hardware changes at all these locations, which is logistically impractical. QUIC shows high potential here due to UDP protocol requiring no changes at intermediate access points. The CUBIC approach specifically is known to be useful for high Bandwidth Delay Product (BDP) networks [14].

Next due course of work is to define a modified FEC feature that is robust to FEC group size with an efficient channel modeling technique that may aid in multiplexing the FEC functionality using free header bits for optimum usage of bandwidth. We also hope to devise a flexible congestion control approach for all BDP scenarios based on the network environment.

V. Conclusions

In an age dominated by video streaming traffic, Google's QUIC protocol finds a critically important place. A primary defining feature of QUIC has been its reduction in data transfer latency, thereby improving response time by removing dependencies in flow management. Testbed results show that overall throughput serviced using QUIC shows an improvement of near 50% over TCP/IP in unrestricted flows for wireless video file downloads. As a result of QUIC's 0-RTT handshake, it achieves a minimal response time (less control overhead) to start a new connection. This results in an improvement of latency parameter of QUIC over TCP/IP for both first time and repeated connections.

Acknowledgment

This work is supported in part by the All India Council of Technical Education (AICTE) under Quality Improvement Program (QIP) and MHRD. Any opinions, findings, conclusions and recommendations expressed are of the author(s) only.

References

[1] Hassan, Mahbub, and Raj Jain, "TCP/IP Fundamentals, TCP/IP Performance over Wireless Networks. *High Performance TCP/IP Networking: Concepts, Issues, and Solutions*", Pearson/Prentice Hall, 2004.

[2] D. Chiu, R. Jain, "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks", *Computer Networks and ISDN systems*, 17 (1), pp. 1-14, 1989.

[3] Ha, Sangtae, Injong Rhee, and Lisong Xu. "CUBIC: a new TCP-friendly high-speed TCP variant." *ACM SIGOPS operating systems review*, 42.5, 64-74, (2008).

[4] Elkhatib, Yehia, Gareth Tyson, and Michael Welzl. "Can SPDY really make the web faster?", *Networking Conference*, 2014 IFIP. IEEE, 2014.

[5] "Hypertext Transfer Protocol version 2 - RFC 7540", Retrieved: Jul., 2017. [Online] Available: https://tools.ietf.org/html/rfc7540

[6] R. Hamilton, J. Iyengar, I. Swett, and A. Wilk, "QUIC: A UDP-Based Secure and Reliable Transport for HTTP/2", [Online], Available: https://tools.ietf.org/html/draft-tsvwg-quic-protocol-02

[7] Megyesi, P., Krmer, Z. and Molnr, S., "How quick is QUIC?," *International Conference on Communications (ICC)*, IEEE, 2016.

[8] "QUIC Internet Draft," Retrieved: Aug., 2017. [Online] Available: https://tools.ietf.org/html/draft-tsvwg-quic-protocol-02

[9] "QUIC Test Server", Retrieved: Aug., 2017, [Online] Available: http://www.chromium.org/quic/playing-with-quic

[10] "Speedtest Networking Monitoring Tool", Retrieved: Aug., 2017. [Online] Available: http://beta.speedtest.net/0

[11] G. Carlucci, L. De Cicco, and S. Mascolo, "HTTP over UDP: an Experimental Investigation of QUIC", *ACM SAC?15*, 2015.

[12] Srivastava, A., "Performance Analysis of QUIC Protocol under Network Congestion", M.S. Computer Science, Worcester Polytechnic Institute, 2017.

[13] S. Cook, B. Mathieu, P. Truong, and I. Hamchaoui, "QUIC: Better For What And For Whom?", *Proceedings of IEEE International Conference on Communications (ICC)*, May 2017.

[14] M.A. Alrshah, M. Othman, B. Ali, et al., "Comparative study of high-speed Linux TCP variants over high-BDP networks", *J. Netw. Comput. Appl.*, 43, 2014.

[15] H. Balakrishnan, V.N. Padmanabhan, S. Seshan and R.H. Katz, "A comparison of mechanisms for improving TCP performance of wireless links", *IEEE/ACM Transactions on Networking*, 6(5), 1997.

[16] Langley, A., Riddoch, A., Wilk, A., Vicente, A., Krasic, C., Zhang, D., Yang, F., Kouranov, F., Swett, I., Iyengar, J. and Bailey, J, "The QUIC transport protocol: Design and Internet-scale deployment", *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pp. 183-196, 2017.