User Experience Evaluation of HTTP/3 in Real-World Deployment Scenarios

Abhinav Gupta and Radim Bartos
Department of Computer Science
University of New Hampshire
Durham, NH 03824, USA
{ag1226,rbartos}@cs.unh.edu

***Abstract*—QUIC, a standard internet protocol, was developed to perform better in high loss and latency networks. HTTP/3 takes the real benefits of QUIC. In this paper, empirical study of Quality of Experience (QoE) under realistic network scenarios along with the impact of local connectivity, server locations and server software is explored between HTTP/3 and HTTP/2 protocols. To explore the QoE, this paper presents the use of throughput and First Contentful Paint (FCP) metric of the Lighthouse, an open source automated tool by Google to measure the user experience and capture performance of the network protocol. The findings indicate that HTTP/3 performs better than HTTP/2 in more challenging network conditions. The experiments also show that while throughput strongly correlates with FCP for HTTP/2, HTTP/3 throughput is not a good predictor of FCP.**

***Index Terms*—HTTP/3, QUIC, QoE, Lighthouse, Protocol Performance**

I. Introduction

Everyday internet traffic is increasing exponentially and so does the need to have a reliable high speed data transfer. Internet service providers are struggling for low latency and high throughput networks as even a 100 ms delay can cause a loss of millions of dollars [1].

Google was preeminent to come up with a new protocol known as QUIC (Quick UDP internet connections) for providing fast data transfer in high loss and latency scenarios. QUIC amalgamates the advantages of both TCP and UDP into one protocol to perform well in high loss and latency scenarios. Google's version of QUIC (gQUIC) was designed and implemented by Jim Roskind [2] in 2012, it was publicly announced in 2013 [3]. gQUIC is mainly deployed at Google's servers and clients such as YouTube and Google's Chrome browser.

IETF recently standardized QUIC in RFC 9000 [4]. Protocols such as HTTP/3, SSH, DNS can be run on top of QUIC [5]. HTTP/3, an application layer protocol, gets the support of streams from within the QUIC protocol [6]. HTTP/3 takes the advantage of QUIC features such as multiple independent streams, Connection ID, frames and zero RTT (Round Trip Time) connectivity. HTTP/3 is in the final stage of standardization by IETF [6]. QUIC was mainly designed to perform better in poor network conditions and provide the user a good Quality of Experience (QoE), which measures user's satisfaction with a service.

The main goal of this paper is to evaluate the QoE of HTTP/3 by comparing the throughput and Lighthouse's First Contentful Paint (FCP) metric among different HTTP/3 and HTTP/2 servers in realistic networks. FCP is an important metric since it indicates how quickly meaningful information is going to appear on the screen.

II. Background

Performance has been an issue since the inception of the World Wide Web. Recently, there has been an increase in latency-sensitive applications and web latency needs to be reduced [7].

HTTP/1 was used for communicating for around 15 years, however due to the issues encountered with HTTP/1, in 2009, SPDY a protocol proposed by Google was later integrated into HTTP/1 and is designated as HTTP/2 [8], which runs on top of TCP. There are impediments of TCP which makes it a slow protocol: (i) long connection establishment time as it can take upto three RTTs with TLS authentication [9] (ii) reliable data transfer which means that even if a single packet is lost then all the data needs to wait for retransmission which leads to Head-of-Line (HOL) blocking. A more efficient protocol was needed for handling today's internet traffic which led to the development of QUIC.

QUIC runs on top of UDP. UDP also known as User Datagram Protocol is a simple protocol with least overhead as compared to other transport layer protocols. UDP has no congestion control of its own. QUIC implements the congestion control in user space instead of kernel space and also helps in reducing the latency. QUIC has faster connection establishment than TCP and can achieve zero RTT connection establishment in case of a previously established connection. In QUIC protocol there can be multiple streams active on a connection at a time. Traffic through QUIC can traverse middle boxes easily as the information is encrypted end to end and there is no threat of information leak [7]. QUIC uses a unique Connection ID, that enables delivery even if there is a change in the IP address. QUIC, a newly standardized protocol acts as backbone to HTTP/3.

Many open-source implementations of QUIC and HTTP/3 have emerged since 2016, which help in exploring the func-

978-1-7281-8688-7/22/$31.00 ©2022 IEEE

tioning of the protocol [10]. Due to the advantages QUIC holds over TCP and its continued wide adoption, resulting in a need to study and compare the HTTP/3 and HTTP/2 QoE in diverse realistic network scenarios.

III. Related Work

There have been a handful of QoE studies on QUIC or HTTP/3 in realistic networks [11]–[14]. Researchers originally started exploring the performance of QUIC by studying the Page Load Time (PLT) metric. However, with the advancement in the internet technology and web-pages the researchers are now shifting the focus towards QoE. Moreover, there is an increased focus in analyzing and benchmarking the performance of HTTP/3.

To the best of our knowledge, the first QUIC performance study was done in 2014 by R. Das [15] on gQUIC. Das et al. evaluated the performance of QUIC by exploring the PLT in QUIC, SPDY and HTTP/1.1 on Alexa's top 500 websites using 100 different network configurations. Author found that in case of very low bandwidth links (0.2 Mbps), HTTP/1.1 performs the best followed by QUIC and SPDY. When the link improves from very low to low bandwidth (0.3–1.0 Mbps), QUIC fared better than HTTP/1.1. Furthermore it was observed that QUIC improved when Round Trip Time increased and fared better than HTTP/1.1 in case of small objects and HTTPS web-pages.

Biswal et al. [16] focused their study on the synthetic web-page load times on gQUIC version 23. They orchestrated the web pages consisting of static objects. Additionally, popular web-sites from the Alexa rankings were also considered. It was observed in their experiment that more than 90% of synthetic web pages loaded faster with QUIC in case of poor network conditions (low bandwidth, high loss and latency). The author also reported that QUIC performance improved with the increase in object size.

Cook et al. [17] performed the experiments in local testbeds as well as on the internet to evaluate the PLT. They observed that QUIC performed better in wireless mobile networks, whereas QUIC's benefits were not obvious in normal network conditions.

Seufert et al. [14] conducted a research to compare the QoE benefits between QUIC and TCP. Video streaming and Web browsing were used for the comparison. QoE factors like PLT, initial delay, visual quality and stalling were considered for observing the QoE to the end user. Authors observed that both the protocols had a similar QoE.

Trevisan et al. [12] did a QoE study on HTTP/3. While only controlling the client, a number of different HTTP/3 websites hosted on the internet were visited. The focus of their study was on performance gains achieved by QoE-metrics. Authors found that HTTP/3 had substantial benefit only in case of high latency and low network bandwidth and HTTP/3 performance relies on the orchestration of server-side platforms.

Another QoE analysis comes from Saif et al. [18]. The paper reports a study comparing QoE between HTTP/3 and HTTP/2 in virtual networks with artificially introduced network impairments. In [18], quantitative throughput is measured and Lighthouse tool is used for measuring the web-page performance. Nginx server with an experimental patch (QUICHE) from Cloudflare was used to provide support for HTTP/3 in their experiment. The authors state that HTTP/3 performed mostly poorly than HTTP/2 but had a higher average throughput for HTTP/3.

The research presented here extends the work [18] to different realistic network scenarios and explores the performance of more HTTP/2 and HTTP/3 servers.

IV. Experimental Setup

In this paper the experiments were designed to capture typical deployment scenarios with respect to local connectivity, server locations and protocols. The following subsections talk about these three factors in more detail.

A. Local Connectivity

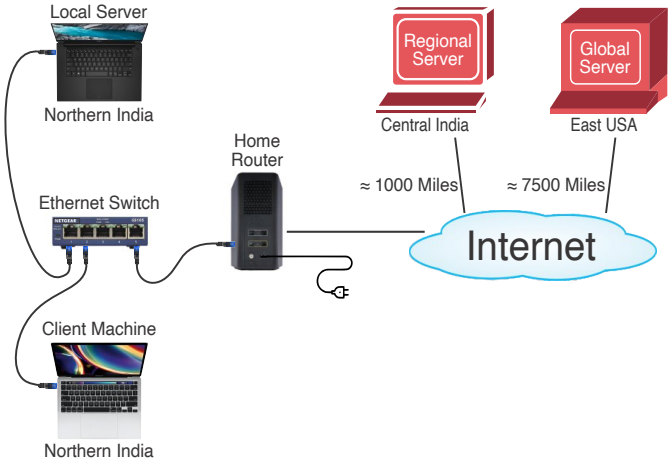

Fig. 1. Wired Experimental Setup

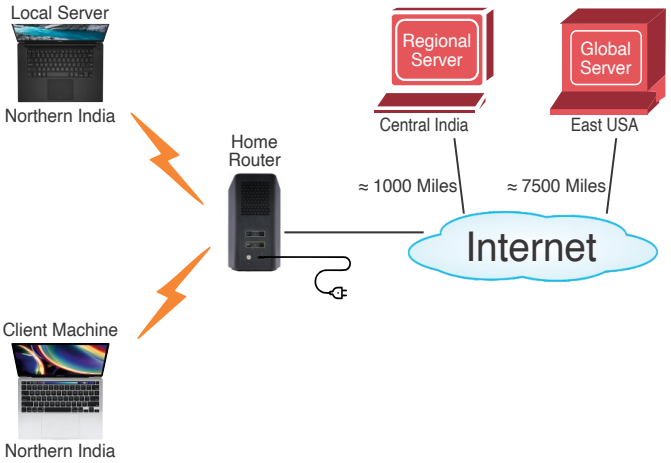

Fig. 2. Wireless Experimental Setup

Figure 1 and 2 show the experimental setup. In the wired setup the client, local server, and the wireless router are connected using a wired Gigabit Ethernet switch. Whereas

TABLE I
SERVER SPECIFICATIONS

	Location	*State, Country*	*Linux Kernel*	*RAM*
Local	LAN	Haryana, India	5.8.0-53-generic	32 GB
Regional	Central India-Azure	Maharashtra, India	5.4.0-1043-azure	8 GB
Global	EAST US-Azure	Virginia, USA	5.4.0-1040-azure	8 GB

in the case of wireless setup the client and local server are connected using a wireless router.

The client machine was located in Northern India. A MacBook Pro laptop running macOS Catalina version 10.15.7 was used as a client machine. Chrome browser version 91.0.4472.114 and Lighthouse tool version 7.3.0 in the client machine were used to carry out the measurements for the throughput and FCP metric of the Lighthouse. Firefox, Safari and Microsoft Edge also support QUIC protocol, however Chrome browser was considered due to maturity of it's QUIC implementation.

B. Server Locations

The servers were hosted in Local, Regional and Global locations. Local server was hosted on a local machine. Regional and Global servers were hosted using Microsoft Azure virtual machines. Regional server is located in Central India and the Global server is in the East USA. Combining the three server locations and two local connectivity options as seen in the previous subsection, we get a total of six scenarios which are referred to as Local Wired, Local Wireless, Regional Wired, Regional Wireless, Global Wired, and Global Wireless.

The Local Wired scenario replicates no impairment scenario. Regional server, which is hosted in Central India, corresponds to typical server location for content transfer to the audience. Global server, which is hosted in the Eastern USA represents a high latency network. *Ping* and *traceroute* utilities were used to calculate the base latency and hop for Regional and Global servers from the client machine. The Regional server exhibited approx. 40 ms latency and 19 hops whereas the Global server exhibited a latency of approx. 230 ms and 24 hops. Client to server physical distance for Global scenario is approximately 7.5 times more than Regional scenario. The choice of these server locations helps in analyzing the protocol behavior in different network settings. Server specifications are shown in the Table I.

C. Server Software

Hypercorn, Nginx, AIOQUIC and QUICHE servers were used in this experiment (see Table II). Hypercorn and Nginx support both HTTP/2 and HTTP/3 protocols, whereas AIOQUIC and QUICHE are HTTP/3 servers. Hypercorn uses AIOQUIC to provide HTTP/3 support. Certificates from AIOQUIC and QUICHE were used for Hypercorn and Nginx. Unlike [18], which used QUICHE from Cloudflare to provide HTTP/3 support to Nginx, this paper uses Nginx's official QUIC implementation [19]. Nginx's own QUIC implementation was preferred because the preliminary experiments performed on the Nginx (QUICHE) showed that the server had stability issues in real networks. When a file was downloaded from Nginx (QUICHE) using HTTP/3 protocol, the file download failed with network error in the browser. A Wireshark trace was used to further examine the issue and it seemed to be caused by an unexpected change in Connection ID. Saif et al. [18] reported results based on Nginx (QUICHE) from a testbed that utilized virtual networks with artificially introduced network impairments. We did not experience the Connection ID issues outlined above in an environment similar to [18]. However, the cause of change of Connection ID in case of real networks remains unresolved. Next section talks about the performance metric used.

TABLE II
SERVER SOFTWARE DETAILS

Server	*Protocol*	*Version*
Iperf3 [20]	TCP	3.9
Nginx-H2 [19]	HTTP/2	Nginx-1.21.1
Nginx-H3 [19]	HTTP/3	Nginx-1.21.1
Hypercorn-H2 [21]	HTTP/2	Hypercorn-0.11.2
Hypercorn-H3 [22]	HTTP/3	Hypercorn-0.11.2, AIOQUIC 0.9.11
AIOQUIC-H3 [23]	HTTP/3	0.9.11
QUICHE-H3 [24]	HTTP/3	0.8.1

V. PERFORMANCE METRICS

Throughput and FCP were used to analyze the QoE observed from different protocols. For the throughput experiments a 50MB file was replicated in each server to maintain the uniformity. The file was downloaded from HTTP/2 and HTTP/3 servers to measure the throughput.

While the throughput signifies the amount of data transferred per unit of time, FCP determines the time it takes for the initial Document Object Model (DOM) content to be loaded on the browser screen [25] and is more oriented towards the QoE. FCP metric was preferred to analyze the QoE of the web-page since PLT alone does not give comprehensive analysis of the QoE. It has also been said in [26] that user experience depends on the overall page load process not just on the precise time at which the page download is complete.

FCP is one of the six performance metrics of the Lighthouse tool. Lighthouse's other performance metrics: (i) Speed Index (SI) (ii) Largest Contentful Paint (LCP) (iii) Time to Interactive (TTI) (iv) Total Blocking Time (TBT) (v) Cumulative Layout Shift (CLS) were also recorded but FCP is of the main interest in this article. Detailed description of Lighthouse

TABLE III
WEB PAGE SPECIFICATIONS

Type	*Total Bytes*	*Number of Files*
HTML	9.36 KB	1
CSS Style Sheets	7.89 KB	3
Total Images	1.23 MB	36
JavaScript	364 KB	20
Total	1.60 MB	60

scoring scheme is provided in the performance metric section of [18]. FCP was given the utmost importance due to the simplicity of the web-page considered.

In order to measure the FCP time, a simple static web-page with images and JavaScript was hosted on different HTTP/2 and HTTP/3 servers. The web-page had no videos or multimedia content. There were no external scripts or content loaded from the web-page. The web-page specifications are given in the Table III. It was preferred to use a static web-page in our study to maintain consistency between experiments and to avoid the effect of dynamic content in the production web-pages. In the next section methodology for the experiment is explained.

VI. METHODOLOGY

The client machine ran the Chrome browser to download the file to measure the throughput and the Lighthouse tool was invoked to measure the FCP metric. Chrome and Lighthouse both support the HTTP/3 protocol which can be enabled by using the flag *--origin-to-force-quic-on=<host>:<port>* at startup through Command-Line Interface (CLI), this flag forces the application to connect to a particular host using QUIC protocol instead of TCP. To avoid having the observations impacted by caching, Chrome browser was started in incognito mode. Lighthouse clears caches by default for every audit.

Lighthouse, when invoked from CLI, uses Chrome Canary or Chrome to perform its audits depending upon the availability of the Chrome binary, with the preference for Chrome Canary if both the binaries are present. Because of this, Lighthouse used Chrome Canary to perform the FCP audits.

A packet trace was captured for throughput measurement using *tshark*. Capinfos was used to obtain the throughput from the trace. Baseline throughput observation was established by Iperf3, which was run for 60 seconds for each server.

Lighthouse FCP metric results were generated in JSON format by passing the *output json* option while running the lighthouse audits through CLI. *jq* is used to parse the JSON output. The runs were fully automated using the shell scripts. Ten iterations of each scenario were measured and then the average value was finally considered to minimize the variance. Results are discussed in more detail in the next section.

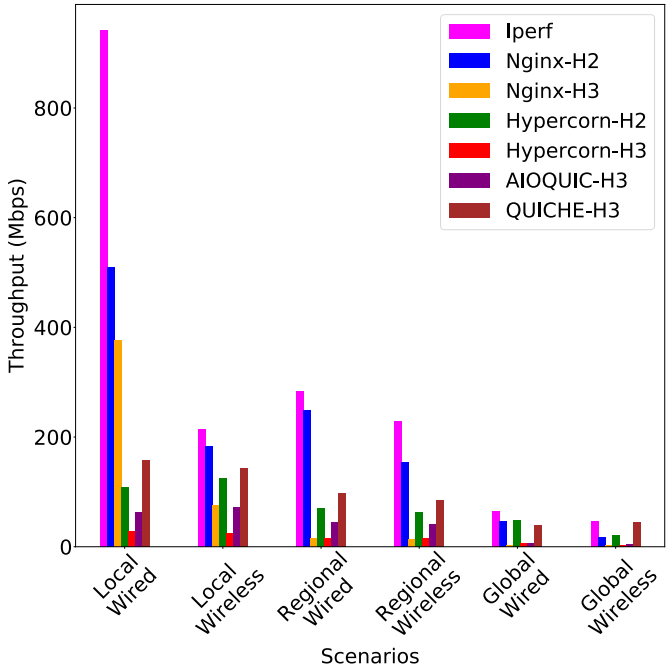

Fig. 3. Throughput for each Scenario

VII. RESULTS

A. Throughput

This section presents the experimental results for the six scenarios and two performance measures: throughput and FCP.

Figure 3 shows the throughput in six different scenarios for each server. Iperf3 results show the baseline measurement of all the servers. It was seen that Nginx-H2 had the highest throughput in Local and Regional scenarios, however QUICHE-H3 had the highest throughput in the Global Wireless scenario. This is consistent with the observation in [16], which also says that QUIC performs better in high loss and latency scenarios.

It is interesting to observe how Nginx-H3 throughput dropped from 376.2 Mbps in Local Wired to 2.5 Mbps in Global Wireless scenario, whereas throughput of other HTTP/3 servers did not dip so drastically. We see that HTTP/2 throughput fared better than HTTP/3 in all the scenarios, with QUICHE being the exception in Global Wireless scenario as said before. Hypercorn-H3 turns out to be the most underperforming server.

Furthermore, Figure 4 and 5 show a comprehensive performance comparison between the servers. Figure 4 presents the same results, in this case comparing the performance of each server for different scenarios. Figure 5 shows throughput of servers relative to Iperf3.

Though throughput gives us a good idea about data transfer speed, it does not give us information about the overall user experience. For better understanding of QoE Lighthouse's FCP time is discussed in the next subsection.

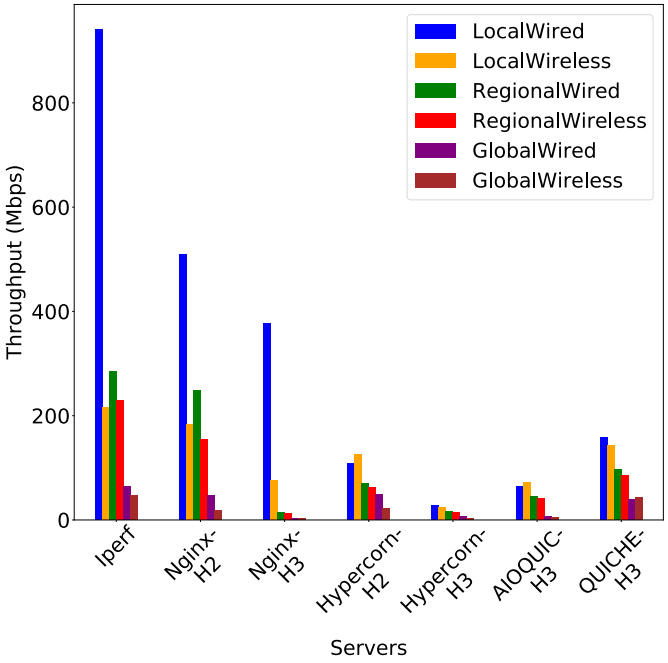

Fig. 4. Throughput for each Server

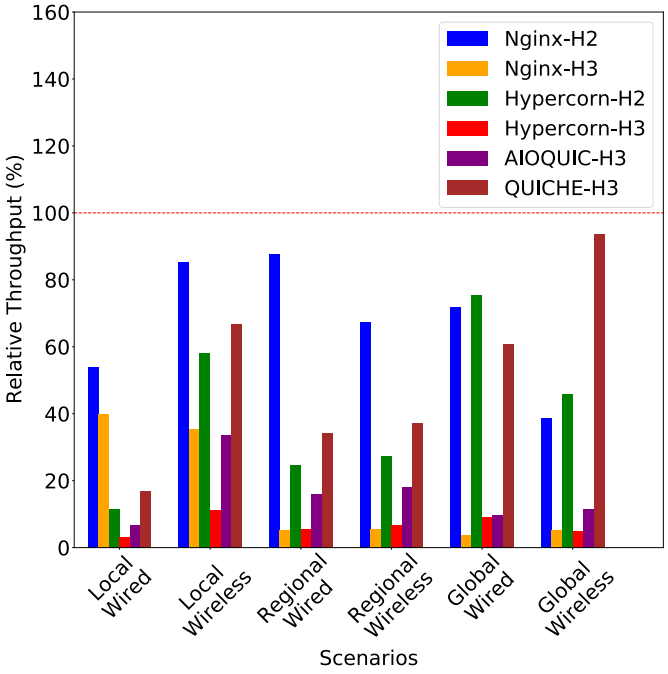

Fig. 5. Throughput relative to Iperf3

B. First Contentful Paint (FCP)

The FCP measure of the Lighthouse is triggered when any render is detected in the browser [27]. Less FCP time indicates that the client starts rendering quickly [28]. FCP time achieved in six different scenarios for each server is plotted in Figure 6, it was observed that Nginx-H2 takes the least FCP time in all scenarios except in case of Global Wireless scenario where Nginx-H3 outperforms all other servers.

AIOQUIC and QUICHE perform similarly in the case of the Local and Regional scenarios, however in Global scenarios their performance is not competitive. Hypercorn-H3 had the worst FCP time in all the scenarios. Figure 7 shows the same FCP results, in this case comparing the time each server takes for different scenarios. It can be easily observed how the FCP time increases for each server from Local Wired to Global Wireless scenario.

Other Lighthouse performance metrics were also recorded. For SI it was observed that Nginx-H2 was always leading, followed by Hypercorn-H2 in most of the scenarios. HTTP/2 largely outperformed HTTP/3 for SI. Nginx-H2 had the best LCP score. HTTP/2 performed better than HTTP/3 for LCP metric. It was seen that TTI scores had a similar trend as FCP scores. TBT was 0 ms for all servers in all scenarios. CLS scores less than 0.1 are considered good scores [29] and it was observed that all the servers had a CLS score of less than 0.1. In the next subsection we compare the HTTP/3 server's performance.

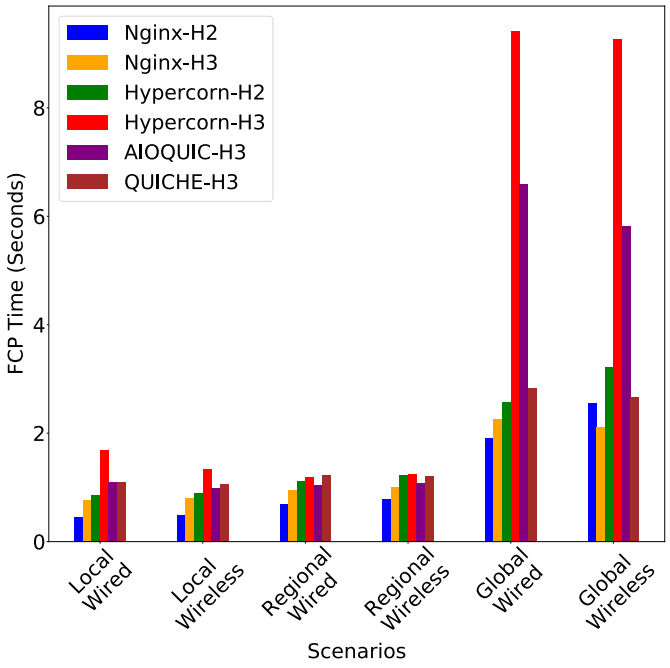

Fig. 6. First Contentful Paint time for each Scenario

C. HTTP/3 Performance Comparison

Four different HTTP/3 servers were considered in this study. It was found that most HTTP/3 servers performed poorly in Local and Regional scenarios. Nginx-H3 performed the best in the Local Wired scenario whereas QUICHE-H3 performed best in the Global Wireless scenario. QUICHE-H3 had a stable throughput as compared to other HTTP/3 servers. AIOQUIC-H3 follows QUICHE-H3 in terms of throughput. Hypercorn-H3 had a poor performance as compared to other HTTP/3 Servers. Nginx-H3 performed well in Local Wired scenario but its throughput sinks in Regional and Global scenarios.

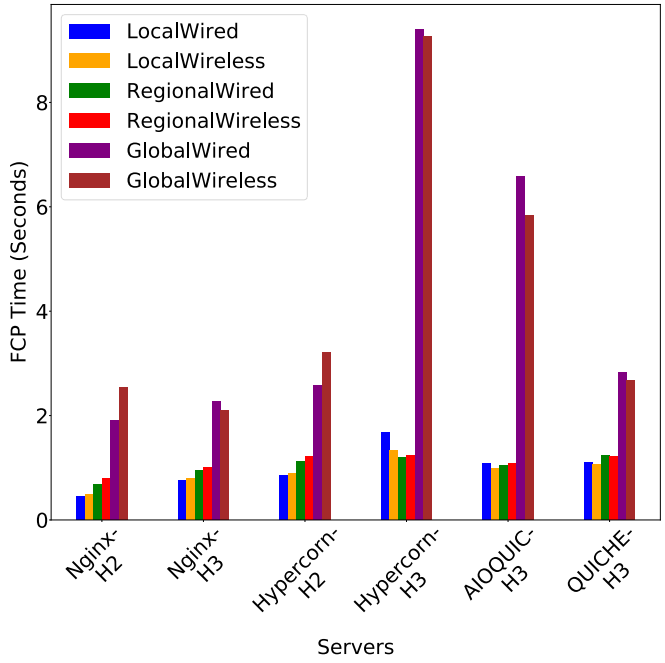

Fig. 7. First Contentful Paint time for each Server

There is a considerable difference in the FCP time among the HTTP/3 servers when it comes to the Global Wired and Wireless scenarios. Nginx-H3 takes the least FCP time, followed by QUICHE-H3 in Global scenarios. However, Hypercorn-H3 and AIOQUIC-H3 do not perform well.

VIII. Conclusion

To understand the performance of HTTP/3 and HTTP/2 protocols, we measured throughput and FCP in diverse scenarios and observed that for HTTP/2 protocol, the servers with high throughput also had high FCP score (low FCP time). Hence, throughput and FCP for HTTP/2 correlate with each other, whereas in case of HTTP/3 protocol no discernible pattern is found. The only exception was found in the case of the Global Wireless scenario where a correlation is observed between throughput and FCP for QUICHE-H3.

It is seen that HTTP/3, which runs on top of QUIC protocol fared better than HTTP/2 protocol in Global Wireless scenario but in Local and Regional scenarios HTTP/2 outperformed HTTP/3.

Though HTTP/2 performed better than HTTP/3 in Local and Regional scenarios, it is also to be noted that HTTP/3 and QUIC implementations are experimental and still getting fine tuned. Whereas, HTTP/2 implementations are highly optimized and are production ready implementations.

Performance gain in HTTP/3 in more challenging network conditions in the Global Wireless scenario seemed to be due to the advanced QUIC protocol design and features, such as improved stream multiplexing to prevent the head of line blocking and optimized congestion control implemented in user space which allows faster handshakes.

To extend the present work more detailed analysis tools will be added so that the root cause of low throughput and FCP scores can be identified. More complex web-pages with videos will be integrated to test the performance of HTTP/2 and HTTP/3 servers. Specifically for HTTP/3, *qlog* [30] integration is planned so that the functioning of various streams can be studied in more detail. Our future research will include QoE analysis of HTTP/3 and HTTP/2 servers in mobile networks and devices, experiments with different browsers as the HTTP/3 protocol, and whether HTTP/3 performance can be further improved by changing the congestion control mechanism of the protocol or by integrating the QUIC protocol into the kernel.

References

[1] Y. Einav, "Amazon Found Every 100ms of Latency Cost them 1% in Sales." [online]. Available at https://www.gigaspaces.com/blog/amazon-found-every-100ms-of-latency-cost-them-1-in-sales/ Accessed: 2021-06-10.

[2] D. Stenberg, ""HTTPS:// URLs"." [online]. Available at https://http3-explained.haxx.se/en/h3/h3-https Accessed: 2021-06-16.

[3] I. Mayersen, "Google Chrome browser is rolling out HTTP/3 via IETF QUIC." [online]. Available at https://www.techspot.com/news/87058-google-chrome-browser-rolling-out-http3-ietf-quic.html. Accessed: 2021-06-10.

[4] J. Iyengar and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport." RFC 9000, May 2021.

[5] R. Marx, "HTTP/3 From A to Z: Core concepts (part 1)." Available at https://www.smashingmagazine.com/2021/08/http3-core-concepts-part1/ Accessed: 2021-10-1.

[6] M. Bishop, "Hypertext Transfer Protocol Version 3 (HTTP/3)," Internet-Draft draft-ietf-quic-http-34, Internet Engineering Task Force, Feb. 2021. Work in Progress.

[7] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar, J. Bailey, J. Dorfman, J. Roskind, J. Kulik, P. Westin, R. Tenneti, R. Shade, R. Hamilton, V. Vasiliev, W.-T. Chang, and Z. Shi, "The QUIC Transport Protocol: Design and Internet-Scale Deployment," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '17, (New York, NY, USA), p. 183–196, Association for Computing Machinery, 2017.

[8] M. Belshe, R. Peon, and M. Thomson, "Hypertext Transfer Protocol Version 2 (HTTP/2)." RFC 7540, May 2015.

[9] K. Nepomuceno, I. N. d. Oliveira, R. R. Aschoff, D. Bezerra, M. S. Ito, W. Melo, D. Sadok, and G. Szabó, "Quic and TCP: A Performance Evaluation," in *2018 IEEE Symposium on Computers and Communications (ISCC)*, pp. 00045–00051, 2018.

[10] I. Q. W. Group, "Implementations." Available at https://github.com/quicwg/base-drafts/wiki/Implementations Accessed: 2021-10-1.

[11] A. Yu and T. A. Benson, "Dissecting Performance of Production QUIC," in *Proceedings of the Web Conference 2021*, WWW '21, (New York, NY, USA), p. 1157–1168, Association for Computing Machinery, 2021.

[12] M. Trevisan, D. Giordano, I. Drago, and A. S. Khatouni, "Measuring HTTP/3: Adoption and Performance," in *2021 19th Mediterranean Communication and Computer Networking Conference (MedComNet)*, pp. 1–8, 2021.

[13] J. Rüth, K. Wolsing, K. Wehrle, and O. Hohlfeld, "Perceiving QUIC: Do Users Notice or Even Care?," in *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, CoNEXT '19, (New York, NY, USA), p. 144–150, Association for Computing Machinery, 2019.

[14] M. Seufert, R. Schatz, N. Wehner, B. Gardlo, and P. Casas, "Is QUIC becoming the New TCP? On the Potential Impact of a New Protocol on Networked Multimedia QoE," in *2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX)*, pp. 1–6, 2019.

[15] S. Das, "Evaluation of QUIC on Web Page Performance." [online]. Available at https://dspace.mit.edu/bitstream/handle/1721.1/91444/893679391-MIT.pdf?sequence=2&isAllowed=y Accessed: 2021-01-16.

[16] P. Biswal and O. Gnawali, "Does QUIC make the Web Faster?," in *2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2016.
[17] S. Cook, B. Mathieu, P. Truong, and I. Hamchaoui, "QUIC: Better For What And For Whom?," in *2017 IEEE International Conference on Communications (ICC)*, pp. 1–6, 2017.
[18] D. Saif, C.-H. Lung, and A. Matrawy, "An Early Benchmark of Quality of Experience Between HTTP/2 and HTTP/3 using Lighthouse," in *ICC 2021 - IEEE International Conference on Communications*, pp. 1–6, 2021.
[19] NGINX, "Experimental QUIC support for nginx." Available at https://quic.nginx.org/readme.html?_ga=2.72641491.1906652836.1626527873-1279281786.1624266205. Accessed: 2021-07-13.
[20] D. J. et.al, "Iperf3." Available at https://github.com/esnet/iperf. Accessed: 2021-02-23.
[21] J. Philip, "Hypercorn." Available at https://pypi.org/project/Hypercorn/. Accessed: 2021-03-23.
[22] J. Philip, "How to serve HTTP 1,2, and 3 in Python." Available at https://pgjones.dev/blog/http-1-2-3-2019. Accessed: 2021-03-23.
[23] L. Jeremy, "AIOQUIC." Available at https://github.com/aiortc/aioquic. Accessed: 2021-02-23.
[24] Cloudflare, "QUICHE." Available at https://github.com/cloudflare/quiche. Accessed: 2021-02-23.
[25] G. Developers, "First Contentful Paint." Available at https://web.dev/first-contentful-paint/. Accessed: 2021-06-18.
[26] D. N. da Hora, A. S. Asrese, V. Christophides, R. Teixeira, and D. Rossi, "Narrowing the Gap Between QoS Metrics and Web QoE Using Above-the-fold Metrics," in *Passive and Active Measurement* (R. Beverly, G. Smaragdakis, and A. Feldmann, eds.), (Cham), pp. 31–43, Springer International Publishing, 2018.
[27] G. M. P. Blog, "First Contentful Paint Explained." Available at https://gtmetrix.com/blog/first-contentful-paint-explained/. Accessed: 2021-06-18.
[28] W. Philip, "First Contentful Paint (FCP)." Available at https://web.dev/fcp/. Accessed: 2021-06-23.
[29] "Getting started with Lighthouse performance audit." Available at https://developers.vtex.com/vtex-developer-docs/docs/vtex-io-documentation-getting-started-with-lighthouse. Accessed: 2021-10-16.
[30] R. Marx, M. Piraux, P. Quax, and W. Lamotte, "Debugging QUIC and HTTP/3 with Qlog and Qvis," in *Proceedings of the Applied Networking Research Workshop*, ANRW '20, (New York, NY, USA), p. 58–66, Association for Computing Machinery, 2020.