On the Impact of the Multiplication Decrease Factor of QUIC

Maiass Zaher
Dept. of Telecomm. and Media Informatics
Budapest University of Technology and Economics
Budapest, Hungary
zaher@tmit.bme.hu

Sándor Molnár
Dept. of Telecomm. and Media Informatics
Budapest University of Technology and Economics
Budapest, Hungary
molnar@tmit.bme.hu

***Abstract*—QUIC is a new transport protocol for transferring web traffic proposed by Google employing CUBIC as the default congestion control algorithm with a small but important modification where it uses a smaller congestion window reduction than the one used in the original CUBIC. Motivated by this change, in this paper we address the robustness and sensitivity of the multiplication decrease factor. More specifically, we investigate the page load time of QUIC as an impact of the multiplication decrease factor in different network environments by changing the loss rate, delay, buffer size, web page size, and bandwidth. We have found that the region where the impact of the change of the multiplication decrease factor in QUIC is pronounced when large web pages are downloaded with low round-trip time achieving 22% decrease in the page load time.**

***Keywords*— QUIC; CUBIC; congestion window reduction; page load time**

I. Introduction

Conventional network stack is the most deployed model for manipulating the network functions. However, one of the reasons which makes the introduction of new solutions in the network difficult is that the network stack resides in the kernel space of the operating systems and it requires a lot of time to adopt changes. This situation motivated network engineers to introduce the network stacks and protocols as a part of the user-space of the operating systems in order to simplify the developing of new solutions. As a result, today there are some user-space network stacks and protocols such as QUIC (Quick UDP Internet Connections) [1], mTCP [2]. Furthermore, the increasingly expanding architectures such as IoT and the cloud computing encouraged network engineers to employ a user-space network stack to mitigate the inefficiency of the conventional general-purpose kernel network stack for providing scalability for service providers. The user-space network stacks have been proven to be developed and configured in a very flexible way [3].

Google proposed QUIC over UDP (User Datagram Protocol) to reduce the latency generated by TCP (Transmission Control Protocol). Google deploys QUIC in its servers and the Chrome browser. Google decided to design a new protocol rather than modify TCP because modifying TCP would take years due to TCP is built in the kernel of the operating system which requires a long life-cycle in the market to adopt. In contrast, QUIC is built in the user-space of the operating system, so it is flexible to deploy and fast to update [4]. Although QUIC employs UDP, QUIC provides a connection-oriented and reliable transfer [5]. However, user-space network protocols and stacks could implement the congestion control algorithms provided for TCP and it encouraged us to study the robustness and sensitivity of the multiplication decrease factor (β) of the CUBIC congestion control algorithm and its impact on the performance of QUIC. In this paper, we investigate the impact of two different values of β of CUBIC algorithm, which is implemented by QUIC as the default congestion control algorithm. Note that, this question is practically also motivated since QUIC changed the congestion window reduction as compared to CUBIC from 30% to 15%. QUIC has been chosen for this research because it is the most well designed and deployed user-space network protocol at the time of writing.

The main contribution of this paper is providing an experimental investigation to check the impact of two different values of β on PLT (Page Load Time) of QUIC in case of a lossy network where we compare the impact of $\beta = \{0.7, 0.6\}$. This paper is organized as follows. Section II presents background of QUIC, CUBIC, the congestion control algorithm employed by QUIC and the related works. In Section III, we present the testbed. Section IV presents the results. Finally, in Section V we give our conclusion.

II. Background and Related work

A. QUIC

Google has developed QUIC to speed up the web traffic more than that with SPDY [6] and Hypertext Transfer Protocol (HTTP)[7]. To achieve this goal QUIC employs UDP as a transport protocol, but at the same time QUIC provides a connection oriented and reliable transmission [4]. QUIC significantly reduces the connection startup delay where it offers *0- RTT* (Round Trip Time) connection establishment if there was a connection established between the client and the server before, as shown in Fig. 1. This *0-RTT* latency for connection establishment is the result of that QUIC applies a dedicated ID to identify a connection instead of using source and destination IP addresses and port numbers. Therefore, the dedicated ID mechanism provides a persistent connection when

This research is supported by Tempus Public Foundation, Stipendium Hungaricum Scholarship Programme and High Speed Networks Lab., Dept. of Telecomm. and Media Informatics, Budapest University of Technology and Economics.

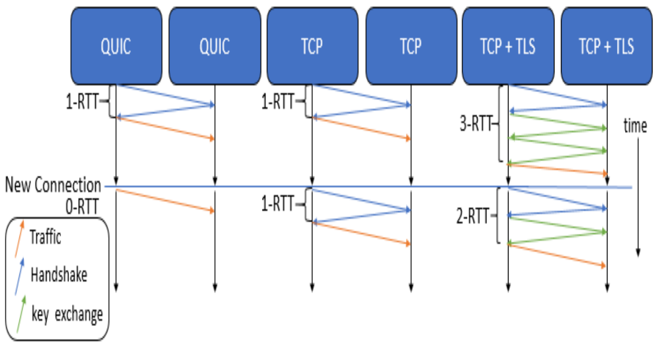

Fig.1. RTT for connection establishment of QUIC, TCP, TCP+TLS

mobile clients change their access points and get a new IP address [5]. In contrast, TCP generates *1-RTT* connection establishment and *3-RTT* in case of using Transport Layer Security (TLS) due to key exchanging, as shown in Fig. 1 [8]. However, QUIC provides features similar to TLS and TCP such as in-order reliable packet transmission, privacy, and congestion avoidance mechanisms, but it presents itself in the network as a UDP connection where it is a tunneling protocol running on top of UDP for multiplexing data streams together. One goal of QUIC is to improve the performance in case of packets loss. To achieve this goal QUIC employs packet pacing to estimate the available bandwidth by tracking the inter-arrivals of packets [5].

B. CUBIC

CUBIC [9] defines the increase rate of the congestion window as a cubic function of the elapsed time since the last congestion event and β which is a coefficient of multiplicative decrease. The dynamics of congestion window are controlled as follows [9].

$$w = C\left(\Delta - \sqrt[3]{\frac{\beta . w_{max}}{C}}\right)^3 + w_{max} \tag{1}$$

Where C is a predefined constant for scaling, Δ is the elapsed time since the last congestion event. w_{max} is the congestion window size just before the last loss event. β is the multiplication decrease factor applied after a packet loss. When a packet loss event occurs the size of the congestion window will be reduced by β as follows [10]:

$$cwnd = cwnd * \beta \tag{2}$$

CUBIC sets β to 0.7 so that the reduction of the congestion window size after a loss event will be 30%. Setting β to a value bigger than 0.5 causes slower convergence which necessitated to add a new mechanism to release a part of the bandwidth hold by existing flows for new flows. Therefore, when a loss event occurs, if the current size of w_{max} is less than w_{max} at the former loss event, CUBIC reduces w_{max} for the current congestion event further to give new flows more time to increase their windows [10].

C. Congestion Control of QUIC

At the time of writing, QUIC implements many TCP congestion control algorithms, but the default is CUBIC. However, QUIC employs packet pacing to estimate the available bandwidth by tracking the intervals between packets at the receiver and the sender [11]. Packet pacing can mitigate the congestion by decreasing the variation in packet flows [8] since a packet loss still represents a sign of congestion in the connection. QUIC reacts to loss events analogously to TCP, specifically, according to CUBIC algorithm. It reduces the sending rate according to the value of β where β equals to 0.7 with considering of emulating n connections, as shown in (3), which reduces the congestion window by 15% instead of 30% in the original CUBIC:

$$cwnd = cwnd * \frac{(n-1+\beta)}{n} \tag{3}$$

However, the intent is to emulate the impact of n connections so that when losing a packet, the streams over one connection have bandwidth equal to n times that of a one connection of traditional TCP to attain flow fairness, where $n = 2$. It is an algorithm known as (Multiple) MulTCP [12][13].

D. Packet Pacing

Packet pacing exists to mitigate the packet loss by sending below than full rate, but it decreases the overall throughput in case of high bandwidth. QUIC applies packet pacing to reduce the page load time where packet pacing tracks inter-packet spacing for estimating the available bandwidth such that a sender cannot send at maximum rate as well as it reduces the packet loss [5]. Experiments conducted by Google have shown that packet pacing has good impact on QUIC performance in the presence of packet loss. However, pacing rate is approximately equivalent to division of the congestion window size over an estimation of RTT [8].

To the best of our knowledge, so far there is no publications regarding the experimental performance study of the impact of β values of QUIC's congestion control algorithm. However, Authors in [14] compared the congestion control dynamics of QUIC CUBIC and TCP CUBIC. They found that QUIC CUBIC has more stable congestion window dynamics than that of TCP CUIBIC because of the congestion window reduction in QUIC is smaller. Miller and Hsiao found that adjusting the value of β of TCP CUBIC resulted in less transfer time in case of packet loss [15].

III. Testbed and Metrics

In this section we describe the testbed and the metrics applied to investigate the impact of two values of β on PLT of QUIC. We employed the testbed, presented in Fig. 2, which consists of two computers. First one is employed as a client and it has Intel Core i5 2.4GHz, 4 GB RAM, Ubuntu 15.10, kernel 4.2.0-22. Second one is acting as a server and it has Intel Core i5 3.4GHz, 8 GB RAM, Ubuntu 14.04, kernel 3.16.0-38. On the server side we used the QUIC server available on Chromium project website [16] to carry out this evaluation, whereas we used Google Chrome v60 as a QUIC client

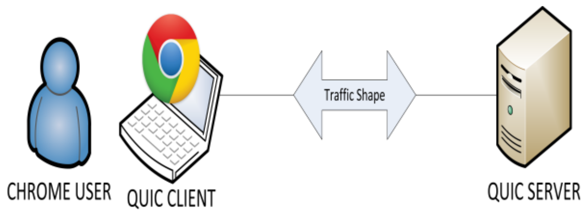

Fig.2. Testbed

which provides QUIC version 38. For emulating different network conditions, we applied *traffic shaping* between the client and the server. We used Netem of the TC (Traffic Control) to emulate different values of bandwidth, loss, queue length, and delay. For loss, we investigated many values of random loss: 0.2%, 0.5%, 1%, 2%, 5% to figure out the impact of β in low, medium and high packet loss conditions where the reduction of the congestion window size will occur after packet loss events. Also, for delay we measured the impact of low and high delay by setting RTT to 5, 10, 200 and 400ms. In all scenarios we set the buffer size equal to the bandwidth-delay product except one where we considered the impact of over-buffered and under-buffered network [17]. Table I presents the parameters applied in the experiment. We applied many combinations of previous parameters to download different web pages from the QUIC server to the QUIC client for both values of β.

We repeated each scenario ten times and computed the average. Furthermore, we investigated the impact of web page size so we conducted a scenario for two different size web pages, namely P = {small, medium} whereas we used the large size web page for the other scenarios [18]. However, all objects of web pages are jpg images only without CSS (Cascading Style Sheets) nor javascript files to eliminate the impact of processing as shown in the Table II. Web pages have higher number of small size objects because QUIC loads small size objects fast [19]. Consequently, these parameters define 45 different scenarios.

We used the developer tool of the Chrome browser which measure the time elapsed since requesting a web page until the page is fully loaded. We disabled the caching in the Chrome browser during our experiment to fetch data from the QUIC server all the time. We had a one connection between the server and the client. Table III summarizes the different values of congestion window reduction according to the investigated values of multiplication decrease factor β.

IV. Results

In this section we present the results of comparing PLT of QUIC under the different scenarios. The goal of this paper is to investigate the impact of applying β = {0.6, 0.7} on the performance of QUIC protocol. We mention that, other values of β resulted in an ordinary behavior such that PLT is better when β is larger than 0.7 and it is worse when β is smaller than 0.7 as also found in previous publications [14][15] except for $\beta = 0.6$. Therefore, we present only the interesting results yielded by applying $\beta = 0.6$ which represent a unique behavior.

We compute the percentage of PLT change, as shown in (4), obtained by β = 0.6 with respect to β = 0.7 which is considered the reference value of the comparison due to the fact that it is the default value of β of QUIC. By tracking the percentage of the PLT change, we evaluate the impacts of different β values. All the following figures show percentages of the PLT change (PLTC) which have been computed by (4).

$$PLTC = \frac{PLT_{0.7} - PLT_{0.6}}{PLT_{0.7}} \times 100 \quad (4)$$

According to (4), the scenarios which have positive *PLTC* indicate that QUIC with β = 0.6 has better PLT by a value equal to the associated percentage in comparison to β = 0.7. On contrary, QUIC has better PLT with β = 0.7 in case of negative *PLTC* in comparison to β = 0.6. We considered three different types of scenarios as shown in Table IV where we did not consider all possible combinations of parameters listed in Table I. Each scenario has been applied five times to evaluate the impact of β under different rates of packet loss. However, we did not consider L = 0% due to the fact that the congestion control algorithm reduces the congestion window size when a loss event occurs only; furthermore, we introduced the packet loss by TC in the direction from the QUIC server to the QUIC client only to avoid the timeout behavior in case of acknowledgement packets are lost. Due to the fact that data flow moves from the server side to the client side only in our experiment, we applied values of β on the server side only.

TABLE I. PARAMETERS USED IN THE EVALUATION

Parameter	**Value**
Bandwidth – B	100, 10 Mbps
RTT – D	5, 10, 200, 400 ms
Packet Loss – L	0.2%, 0.5%, 1%, 2%, 5%
Buffer length – Q	RTT * BW
multiplication decrease factor – β	0.6, 0.7

TABLE II. WEB PAGE STRUCTURE

Web page size - *P*	**Number of objects**		**Size of an object in KB**
Large: 3.3 MB	Small	153	15
	Large	8	135
Medium: 1 MB	Small	50	15
	Large	2	135
Small: 300 KB	Small	11	15
	Large	1	135

TABLE III. VALUES OF MULTIPLICATION DECREASE FACTOR

β	**Congestion Window Reduction**	**Description**
0.7	15%	QUIC default multiplication decrease factor
0.6	20%	5% larger than QUIC's default congestion window reduction

TABLE IV. SCENARIOS USED IN THE EVALUATION

Scenario	**Goal**
Delay impact	Presents the impact of β with respect to different values of D and L
Buffer impact	Presents the impact of β with respect to different values of Q, 2*Q, Q/2 and L
Page size impact	Presents the impact of β with respect to different sizes of P and L

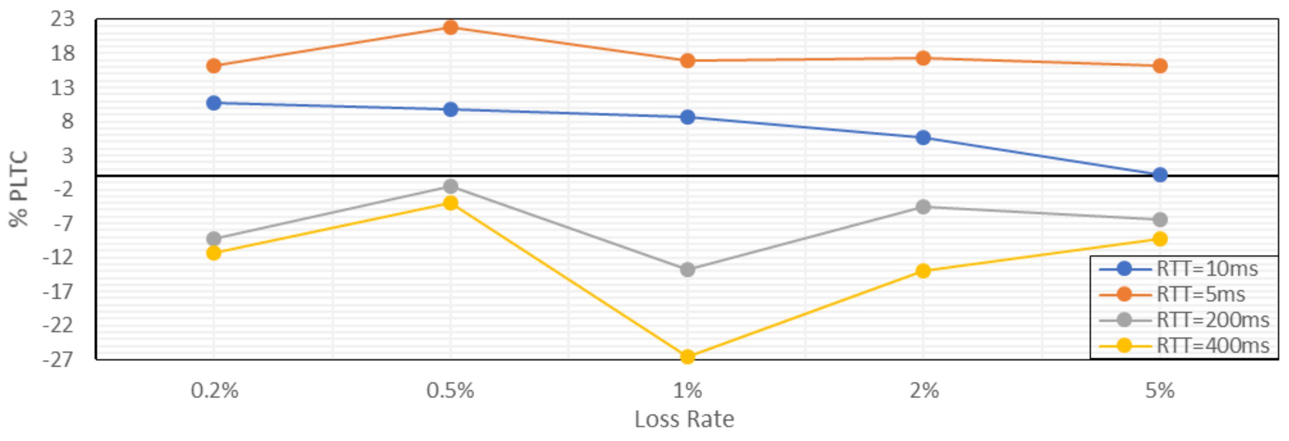

Fig.3. PLTC with respect to delay

A. Delay Impact

In this section we evaluate the impact of β on PLT of QUIC under different RTT values. We considered B = 100 Mbps, D = {200, 400, 10, 5}ms, and P = 3.3 MB. Each combination of D, B, P, β has been applied five times to evaluate the PLT under all different values of L. Buffer size has been set to bandwidth-delay product in this scenario. Fig. 3 presents *PLTC* when β = 0.6 in comparison to β = 0.7 under different loss rates. Fig. 3 shows that all *PLTC* of QUIC with β = 0.6 are positive in case of RTT = 10ms. This is due to the fact that when the network is congested, packet pacing estimates a larger data transmission rate in case of low RTT and the congestion window reduction is 20% than that in case of the congestion window reduction is 15%. This behavior has been approved by repeating same scenario but with smaller delay, as shown in Fig. 3, where RTT has been set to 5ms resulting in PLT better than that when RTT = 10ms. To investigate the impact of high RTT, we performed the scenario with high RTT = {200 ,400}ms as shown in Fig. 3 which shows consistent results that PLT becomes higher than that with β = 0.7 when RTT is high. Consequently, in case of β = 0.6, the PLT will be better up to 21.9% when RTT is 5ms and up to 10.73% when RTT is 10ms than that in case of β = 0.7.

B. Page Size Impact

In this section we compute *PLTC* of β = 0.6 in comparison to that when β = 0.7 to investigate the impact of web page size in both cases under different loss rates. We considered B = 10 Mbps, D = 10 ms and P = {Medium, Small}. We did not consider the large web page for this scenario because it has been applied in the other scenarios. Fig. 4 presents the *PLTC* in case of β = 0.6 under different loss rates in comparison to β = 0.7. It consists of two curves for the two different web page sizes. Fig. 4 shows that β = 0.6 provides better PLT than the case when β = 0.7 under all loss rates for medium web page. This mainly due to the fact that when the network is lossy, smaller reduction in congestion window causes more packet loss when more data needs to be transferred and consequently the overall situation will be worse. Fig. 4 shows that, when medium size web page needs to be transferred in a lossy network, the PLT can be reduced to 8% by applying β = 0.6 in comparison to β = 0.7.

C. Buffer Impact

In this section we measure the impact of buffer size Q when applying two different values of β. We considered B = 100 Mbps, D = 5 ms and P = 3.3 MB for this scenario. Three different buffer sizes have been considered Q = {64, 128, 32}KB with Token Pocket filter. In particular, for Q = 64KB the network is well-buffered, for Q = 128KB the network is over-buffered and for Q = 32KB the network can be considered under-buffered. We applied this scenario five times to consider the different values of loss rate, as shown in Fig. 5 which presents the impact of different buffer sizes. The main observation is that when β = 0.6, QUIC needs less time to load the large web page than that in case of β = 0.7 regardless of the buffer size. It is due to the fact that we have a one connection between the QUIC client and the QUIC server so there are not a lot of data to be transferred; consequently, the receiver's buffer cannot be overwhelmed. The results of this scenario consistent with the results of previous scenarios where the delay is low and the size of web page is large, i.e. the amount of data needed to be transferred is not small, which yields better PLT by a value between 12% - 22% in case of β = 0.6 in comparison to β = 0.7 under different loss rates.

Furthermore, we summarize our measurement results in Table V where we present the count of the positive *PLTC* under the five loss rates of each scenario which implies that QUIC with β = 0.6 has better PLT in comparison to that one with β = 0.7. Table V shows that QUIC with β = 0.6 can provide better PLT under different loss rates in particular when RTT is low and the size of web pages need to be downloaded is not small. However, Table V shows that QUIC with β = 0.6 has better PLT in most of the cases even that it has larger window reduction. It is worth to mention that we established a one connection between the QUIC server and the QUIC client in all scenarios for maintaining the accuracy of results due to the fact that in the time of writing there was not any reliable QUIC server available for evaluating the impact of many concurrent connections.

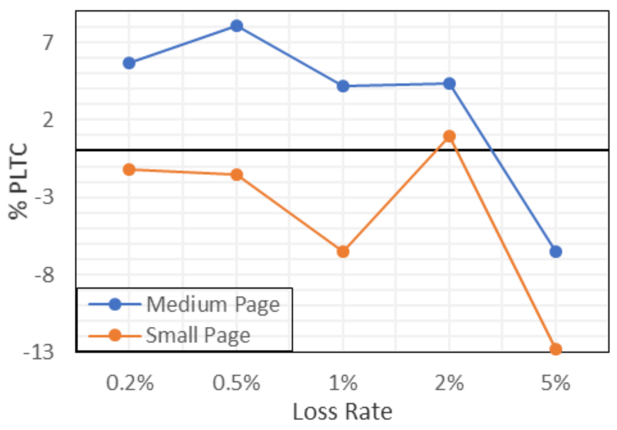

Fig.4. PLTC with respect to page size

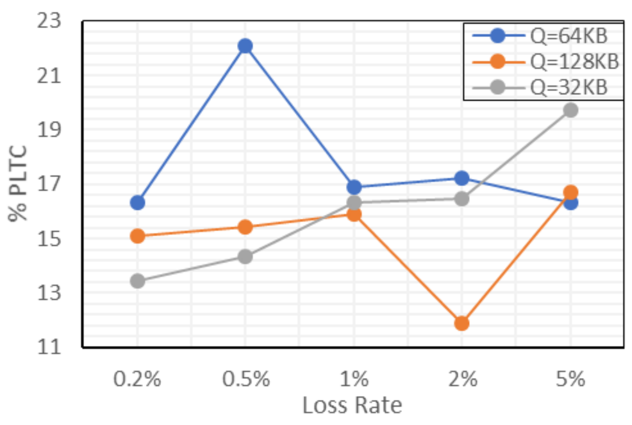

Fig.5. PLTC with respect to buffer size

TABLE V. NUMBER OF POSITIVE %PLTC FOR EACH SCENARIO

Scenario	**Scenario Parameter**	**Number of Cases in which PLT is Improved**
Page Size	Medium	4/5
	Small	1/5
Delay	200ms	0/5
	10ms	5/5
	400 ms	0/5
	5 ms	5/5
Buffer Size	well	5/5
	over	5/5
	under	5/5
Total Number		30/45

V. CONCLUSION

In this paper we presented an experimental investigation of the impact of the multiplication decrease factor used in CUBIC congestion control algorithm applied by QUIC. Under different network conditions, we have measured the percentages of page load time change (PLTC) when setting β of QUIC so that the congestion window reduction will be 20% instead of 15%. We have found that QUIC loads web pages faster (12% - 22%) when $\beta = 0.6$ in comparison to $\beta = 0.7$ in case of large size web page along with small round-trip time whereas we have found that $\beta = 0.7$ provides better PLT when the size of the web page is small and RTT is high. However, when setting $\beta = 0.6$, QUIC has better PLT in most cases. In this study, we present the impact of two values of β under different network conditions in case of one connection between the QUIC client and the QUIC server for maintaining the accuracy, but our future plan is to conduct a research to evaluate the performance of QUIC when there are many concurrent connections at the server side and to compute the value of β by a dynamically adaptive manner.

REFERENCES

[1] 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.

[2] E. Jeong et al.,"mTCP: a highly scalable user-user-level TCP stack for multicore systems," in 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14) , 2014.

[3] I. Marinos, R. Watson and M. Handley, "Network stack specialization for performance," ACM SIGCOMM Computer Communication Review, vol. 44, no. 4, 2014, pp. 175-186.

[4] J. Roskind, "Quick UDP internet connections multiplexed stream transport over UDP," IETF-88 TSV Area Presentation, 2013.

[5] J. Roskind, "QUIC design document and specification rationale," 2013. [Online]. Available: https://docs.google.com/document/d/1RNHkx_VvKWyWg6Lr8SZ-saqsQx7rFV-ev2jRFUoVD34/mobilebasic.

[6] "SPDY protocol – draft 3," [Online]. Available: https://www.chromium.org/spdy/spdy-protocol/spdy-protocol-draft3

[7] "Hypertext Transfer Protocol Version 2 (HTTP/2)," IETF RFC 7540, [Online]. Available: https://tools.ietf.org/html/rfc7540

[8] J. Roskind, "Multiplexed stream transport over UDP," Google, 2013.

[9] S. Ha, I. Rhee, and L. Xu, "CUBIC: a new TCP-friendly high-speed TCP variant," ACM SIGOPS Operating Systems Review, vol. 45, no. 5, 2008, pp. 64-74.

[10] I. Rhee et al., "CUBIC for fast long-distance networks," IETF Internet Draft, draft-ietf-tcpm-cubic-07, 2017.

[11] L. De Cicco, G. Carlucci and S. Mascolo, "Understanding the dynamic behaviour of the google congestion control for rtcweb," in Packet Video Workshop 2013, San Jones, USA, 2013.

[12] F. Kuo and X. Fu, "Probe-aided mulTCP: an aggregate congestion control mechanism," *ACM SIGCOMM Computer Communication Review,* vol. 38, January 2008, pp. 19-28.

[13] A. Langley et al., "The QUIC transport protocol: design and internet-scale deployment," In Proceedings of the Conference of the ACM Special Interest Group on Data Communication, pp. 183-196. ACM, 2017.

[14] G. Carlucci, L. De Cicco and S. Mascolo, "HTTP over UDP: an experimental investigation of QUIC," in 30th Annual ACM Symposium on Applied Computing, 2015.

[15] K. Miller and L. Hsiao, "TCPtuner: congestion control your way," arXiv preprint arXiv:1605.01987, 2016.

[16] Chromium project. https://cs.chromium.org/chromium/src/net/quic/

[17] G. Appenzeller, j. Keslassy and N. McKeown, "Sizing router buffers," in ACM SIGCOMM '04, Portland, USA, 2004.

[18] "HTTP archive," [Online]. Available: http://httparchive.org/interesting.php. [Accessed: 1 November 2017].

[19] P. Megyesi, Z. Krämer and S. Molnár, "How quick is QUIC?," in *Communications (ICC), 2016 IEEE International Conference*, 2016.