Does QUIC Suit Well with Modern Adaptive Bitrate Streaming Techniques?

Abhijit Mondal, *Student Member, IEEE*, Sandip Chakraborty, *Member, IEEE*

***Abstract*—This letter provides a thorough analysis of various recent adaptive bitrate streaming (ABR) techniques over Google's QUIC protocol. More specifically, we focus on the recent developments of the ABR techniques using control theoretic and reinforcement learning based approaches, and compare their performances over HTTP/TCP and HTTP/QUIC in terms of a wide range of video quality of experience (QoE) parameters. We observe that these ABR techniques are more compatible with HTTP/TCP compared to HTTP/QUIC, and the features of QUIC demands for new ABR technique which can balance the three fundamental QoE parameters – bitrate, smoothness and rebuffering during a video playback.**

***Index Terms*—DASH, ABR, QUIC, TCP**

I. Introduction

Quick UDP Internet Connection (QUIC) [1] has been developed and experimentally deployed over the Internet by Google to replace TCP as the transport layer protocol, while addressing the limitations of TCP for end-to-end connection managements over a wide range of applications. While majority of the global Internet traffic originates from various video streaming applications, QUIC claims to reduce the YouTube rebuffering by a margin of 18% for desktop users and 15.3% for mobile users [1]. As Dynamic Adaptive Streaming over HTTP (DASH) [2] has been a de facto for Adaptive Bitrate Streaming (ABR) over the Internet, it would be interesting to explore how QUIC performs over various ABR techniques proposed in the recent literature in terms of end users' quality of experience (QoE).

In DASH, the videos are divided into small segments, and every segment is encoded in multiple quality levels (bitrates). The DASH client measures the network condition and requests for a video segment with the most suited quality level based on the current network condition. The network measurement and the corresponding bitrate adaptation algorithm can be tuned in DASH, and a large number of works have been proposed in the literature to select the optimal ABR technique, such as buffer based (BOLA [3]), using control-theoretic approach (MPC [4]), or based on deep reinforcement learning (Pensieve [5]). However, these existing approaches do not look into the impact of the underlying transport protocols on the video streaming QoE performance. While TCP uses multiple socket connections to download the video and the corresponding audio data, QUIC multiplexes the audio and the video streams over a single UDP socket to download the entire data from the server. As the ABR techniques depend on the channel throughput estimation at the client, such protocol changes are likely to impact the streaming performance.

With the above context, this letter evaluates and compares the performance of various ABR techniques over TCP and QUIC. We develop a testbed setup with the help of a standard DASH player from the DASH Industry Forum, where we integrate the DASH player with QUIC and use emulated network environment based on a large pool of pre-collected network traffic traces. With a total of 45 hours of streaming video data, we have computed three QoE metrics – (a) average playback bitrate, (b) total rebuffering duration, and (c) playback smoothness, while streaming over both TCP and QUIC. Our thorough experiments and observations indicate that recent ABR techniques provides better QoE over TCP compared to QUIC. We investigate further to understand the protocol-level behavior of QUIC, which impacts the QoE performance. Our analysis reported in this letter can help the community to tweak the QUIC configurations to obtain the best QoE performance from the modern ABR techniques.

II. Related Work

Various recent studies [6], [7] have revealed that QUIC can improve the web performance by reducing the page load time even at poor network conditions. Among them, a few works have explored the adaptive streaming performance over QUIC. In [8], the authors have empirically shown that DASH suffers over QUIC. Although they have given the first indication that the current ABR techniques might not perform well over QUIC, their analysis is mostly focused on buffer-based ABR and does not look into various QoE metrics as explored in the recent literature [4], [5]. In a follow-up work [9], the authors have explored the QUIC retransmissions to improve the buffer-based ABR over DASH. In [10], the authors have used an emulated setup to analyze the buffer-based ABR techniques over QUIC and also proposed a QoE prediction mechanism for adaptive streaming over QUIC. Also, these existing studies have indicated that QUIC might not suit well for buffer-based ABR. They have not explored the performance of advanced ABR techniques over QUIC. We argue that there is a requirement to analyze the recent ABR techniques like MPC [4] and Pensieve [5] over QUIC because these recent studies have indicated that buffer-based techniques are aggressive towards video-bitrate maximization whereas suffers in terms of playback smoothness and rebuffering. To the best of our knowledge, this is the first study that explores the performance of advanced ABR techniques over QUIC as the end-to-end transport protocol.

A. Mondal and S. Chakraborty are with Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, India 721302. Email: abhimondal@iitkgp.ac.in, sandipc@cse.iitkgp.ac.in

Manuscript received ...; revised August ...

III. Experimental Setup

For a fair comparison between the performance of DASH over TCP and QUIC, we use an experimental test network as shown in Fig. 1. We use a streaming server to keep all the videos encoded at different video bitrates as mentioned later and a web-based javascript DASH player provided by DASH Industry Foundations (DASHIF)[1] to stream the videos at the client side. To emulate realistic traffic behavior over the test setup, we use a benchmark traffic shaper `Mahimahi` [11] and have created `Mahimahi` compatible traces from two publicly available datasets – (i) a broadband trace from FCC [12] and (ii) the 3G/HSDPA mobile dataset collected in Norway [13].

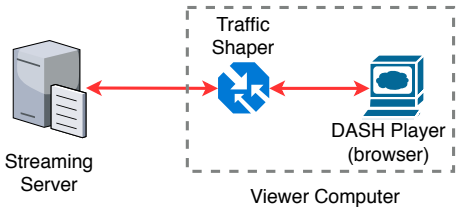


Fig. 1. Experimental setup for video streaming using DASH

We run two servers with the different end-to-end data transfer protocols, one for QUIC and another for TCP. For QUIC server, we use the `golang` implementation of QUIC – `GO-QUIC`[2] (release version 12.0). We have also tested with other QUIC implementations like LiteSpeed QUIC [3] and have similar observations as reported in this paper. We give the results corresponding to the `GO-QUIC` implementation as it has been used by majority of the existing works in literature. For TCP based server, we use the standard `webfsd`[4] web-server (version 1.21). As only the `Google Chrome` and the `Chromium` browsers support the QUIC protocol, we use the `Google Chrome` browser of compatible version 68, for our experiments to stream the videos at the client side over the DASHIF streaming player.

In all the experiments, we use off-the-shelf applications and run them in a non-root user mode. The streaming server and the DASH players run over systems with 8GB of RAM and Intel(R) Core(TM) i5-4590 CPU @ 3.30GHz processor running Ubuntu 16.04.6 LTS operating system on top of Linux 4.9.78. For DASH client, we modify dash.js v2.9.3 to add support for advance ABR algorithms like Pensieve and MPC.

TABLE I
Bitrate and resolution map of DASHified videos.

Bitrate(kbps)	200	400	600	800
Resolution	320x180	320x180	480x270	640x360
Bitrate(kbps)	1000	1500	2500	4000
Resolution	640x360	768x432	1024x576	1280x720

We use a total of 45 hours of videos (about 50 different videos) which have been dashified (encoded in different bitrates as shown in Table I) using the `ffmpeg` tool in eight different video quality levels and two different audio quality levels. We play every videos for all the ABRs and protocol combinations, resulting ≈ 19 days of video playback time. We also maintain the similar playback environment for DASH/QUIC and DASH/TCP for each of the video and the ABR combinations. We have compared the performance across five different ABR mechanisms, namely, BOLA (B) [3], the standard buffer-based quality adaptation of DASHIF (BB), the two variants of model predictive control (MPC) driven approaches [4] – MPC-Fast (MF) and MPC-Robust (MR) and the deep learning driven quality adaptation algorithm – Pensieve (P) [5]. To measure the overall QoE of the video playback, we have used a combined metric of the average quality level, the rebuffering time and the playback smoothness, as used in [4], [5].

$$QoE = \sum_{n=1}^{N} q(R_n) - \mu \sum_{n=1}^{N} T_n - \sum_{n=1}^{N-1} \left| q(R_{n+1}) - q(R_n) \right| \quad (1)$$

Eq. (1) indicates the QoE metric to measure the overall QoE of a video playback session. Here, N is the total number of chunks for the playback video. R_n and $q(R_n)$ are the representation of the playback bitrate of the chunk n and the quality perceived by a user for that chunk n. T_n is the rebuffering time for the chunk n. μ is a QoE weight factor [4]. We consider a linear representation $q(R_n) = R_n$, similar to [4], indicating that the playback quality increases linearly with the increase of playback bitrate.

IV. Results and Analysis

We first look into the overall playback video quality and then dig into the details of individual QoE metrics. As the data used for this analysis have been collected from realistic experiments, we, therefore, avoid any underlying parametric assumption while checking for the statistical significance of the results. Subsequently, we apply Mann–Whitney U test [14] with non-parametric assumptions on all the metrics and report if the results are statistically significant or not with the hypothesis that DASH/TCP works significantly better than DASH/QUIC[5]. We also run a two-way test to check the alternate hypothesis that DASH/QUIC works significantly better than DASH/TCP.

A. Average Video Bitrate

The distributions of the average playback bitrate for the five ABR mechanisms are shown in Fig. 2. For BOLA, we observe that the average video bitrate for DASH/TCP is higher than that of DASH/QUIC. It is known from the existing literature that buffer-based ABR mechanisms aggressively use the highest quality levels, which we also observe in Fig. 2. However, we observe that DASH/TCP always performs better than DASH/QUIC. Indeed, the state-of-the-art reinforcement learning based ABR mechanism, Pensieve, provides much better quality level on top of TCP in comparison to QUIC.

[1]https://github.com/Dash-Industry-Forum/dash.js, (Last accessed: May 17, 2020)

[2]https://github.com/lucas-clemente/quic-go (Last accessed: May 17, 2020)

[3]https://github.com/litespeedtech/lsquic (Last accessed: May 17, 2020)

[4]https://www.gsp.com/cgi-bin/man.cgi?section=1&topic=webfsd (Last accessed: May 17, 2020)

[5]We use the notation "DASH/TCP" to indicate DASH over TCP; similarly "DASH/QUIC" indicates DASH over QUIC.

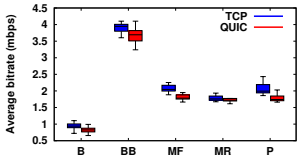


Fig. 2. Average Playback Video Quality for Different ABR Techniques ($p < 0.05$ for all the metrics)

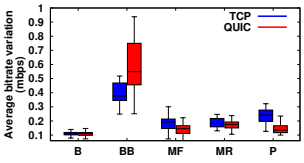


Fig. 3. Average Playback Quality Variation for Different ABR Techniques ($p < 0.05$ for all the metrics except BOLA and MPC-Robust)

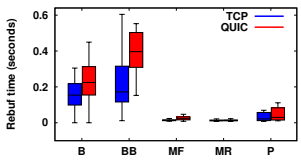


Fig. 4. Rebuffering Time for Different ABR Techniques ($p < 0.05$ for all the metrics except Pensieve and MPC-Robust)

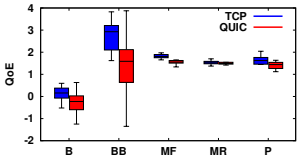


Fig. 5. Overall QoE for Different ABR Techniques ($p < 0.05$ for all the metrics except MPC-Robust)

For all the five ABR methods, we observe that the p-value is less than 0.05 for the Mann–Whitney U test, indicating that DASH/TCP performs significantly better than DASH/QUIC in terms of average playback quality.

B. Quality Level Fluctuation – Playback Smoothness

Next, we observe the average fluctuation in the playback quality levels, which has been shown in Fig. 3. A fluctuation in the quality level indicates less smoothness in the video playback, and, therefore, reduces the QoE. We observe that the differences in average quality level fluctuations between DASH/TCP and DASH/QUIC for BOLA and MPC-Robust are statistically insignificant. However, the figure indicates that quality fluctuation with DASH/QUIC is significantly more for buffer-based ABR, whereas less for MPC-Fast and Pensieve-based ABR, with p-value less than 0.05. This is an interesting observation as we see that the advanced ABR techniques, such as MPC-Fast and Pensieve, provide better playback smoothness with DASH/QUIC, although the supported playback quality is lower compared to DASH/TCP.

C. Rebuffering Time

Fig. 4 indicates the rebuffering time for different ABR techniques over DASH/TCP and DASH/QUIC. We observe that rebuffering is significantly more with DASH/QUIC for BOLA, buffer-based and MPC-Fast, whereas the differences in rebuffering are statistically insignificant for MPC-Robust and Pensieve. For the buffer-based ABR which aggressively download the videos at the highest playback quality, rebuffering is comparatively very high with DASH/QUIC. MPC-Robust and Pensieve show minimal rebuffering, so the differences between DASH/TCP and DASH/QUIC are not statistically significant. We observe that the claim from Google [1] that QUIC enables less rebuffering does not hold for all the ABR techniques and is very much specific to which ABR technique is adopted at the playback client.

D. Overall QoE

Finally, we look into the overall QoE computation as shown in Eq. (1). The overall QoE measurements for all the ABR techniques are shown in Fig. 5. In this experiment, we plot the linear variants of $q(R_n)$ as discussed in the previous section. Our observations from these results are as follows. DASH/TCP provides significantly better QoE compared to DASH/QUIC for all the ABR algorithms except MPC-Robust where the result is statistically insignificant. This indicates that the recent advanced ABR techniques like MPC and Pensieve are more compatible with TCP than QUIC. Indeed from representations of $q(R_n)$, we can say that the above observations are generic across a wide variation of QoE measurements.

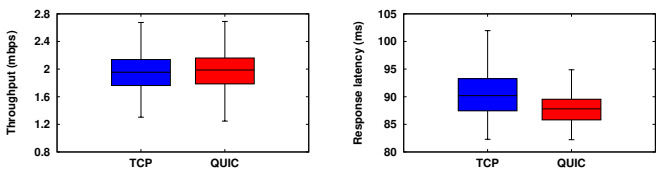
V. Looking Under the Hood

Our thorough experiments over a wide range of ABR techniques indicate that DASH/QUIC does not perform as well as it is supposed to. To explore this further, we perform a set of experiments to understand the issues with QUIC, which affect DASH-based video streaming.

A. Does QUIC Perform Poorly in All Scenarios?

One immediate question that might arise in this context is whether the observed performance drops are generic for QUIC, or whether certain features of QUIC affect the performance of DASH. With the similar network setup as discussed before, we compute the overall throughput and the response latency for TCP and QUIC over two different scenarios – (a) downloading of large web-objects and (b) DASH video streaming.

1) Performance during HTTP Object Downloads: To see how the performances of TCP and QUIC fare during the download of HTTP web objects, we perform a set of experiments over the same network setup with emulated bandwidth control via `Mahimahi` traffic shaper. We download HTTP web objects (HTML and data files) of predefined sizes (from 2MB to 50MB) for 40 sequential HTTP requests in each session. We give a pause of 500ms before requesting the next web object. We repeat each session three times for both TCP and QUIC.

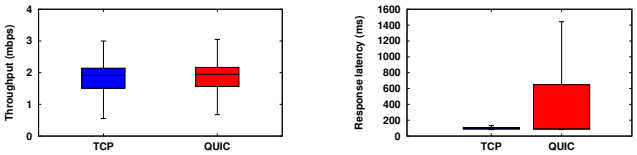


(a) Throughput (b) Response latency

Fig. 6. TCP and QUIC performance during HTTP web object download

Fig. 6a shows the distribution of throughput observed against each web object downloads, while Fig. 6b indicates

the response latency observed by each HTTP request. The throughput is computed as the object size divided by the download time where the download time is the time difference between the first and the last bytes received for that web object. The response latency is computed as the time between the initiation of the HTTP request and the time when the first byte of the response is received. From the figures, we see that the throughput is similar for HTTP/TCP and HTTP/QUIC. The response latency for HTTP/QUIC is slightly lower than HTTP/TCP. These experiments shows that QUIC performs better than TCP in terms of response latency which is an important QoE metric for web object downloads.



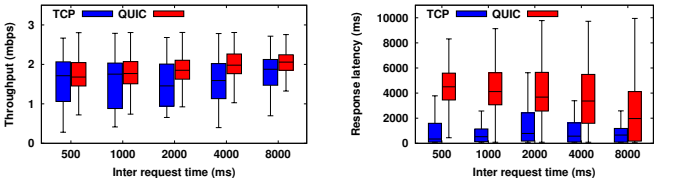
(a) Throughput (b) Response latency

Fig. 7. TCP and QUIC performance during DASH video segment download

2) Performance during Video Streaming: Fig. 7a shows the distribution of the throughput observed during the video playback, combining the data from all the five ABR mechanisms. A statistical test also indicates that the difference between TCP and QUIC in terms of throughput is not significant. Interestingly, the predicted throughput during video streaming is one of the important metrics used by all the ABR mechanisms for deciding the optimal bitrate. Fig. 7b plots the distributions of the response latency, combining the videos from all the five ABR mechanisms. We have an interesting observation here – although the difference in the median of the response latency for DASH/QUIC and DASH/TCP is not significant, the upper quartile for the response latency of DASH/QUIC is significantly higher than the upper quartile of DASH/TCP. This indicates that with QUIC, the response latency sometime becomes very high – this observation is opposite to what we have observed in Fig. 6b. It can be noted that the throughput computation does not consider the response latency, rather it considers the time difference between the first and the last bytes received for a video segment. The ABR algorithms primarily select the bitrate based on the computed throughput for the last few video segments. If the computed throughput is high, the DASH client requests for the next video segment in an increased quality level. However, if the response latency is high, this segment may take longer time to reach the client, resulting in a rebuffering and subsequent drop in the quality levels for the next video segments. We do not see this problem for TCP, as the response latency correlates with the computed throughput; however, this correlation does not hold for QUIC. Next, we dig further to find out the reason behind the high response latency observed during the video streaming using QUIC.

B. Exploring TCP and QUIC Connections during a Video Streaming

During the dashification of a video with an embedded audio, the standard practice is to first segregate and segment the video data and the corresponding audio data, and then encode the video and the audio segments separately in their respective available encoding formats. Now the DASH client creates two different HTTP streams for downloading the video segments and the corresponding audio segments. For DASH/TCP, two different TCP sockets are created between the DASH client and the server for these two HTTP streams, whereas for DASH/QUIC, both the HTTP streams are multiplexed, and the HTTP messages are exchanged over a single UDP socket. This brings the next question – how do TCP and QUIC fare for two parallel but interdependent application streams between the same client and the server? To answer this question, we do the next experiment over the same network setup as discussed before. In this experiment, we create two HTTP streams where both the streams request for the HTTP objects (files) in parallel. The object sizes are varied from 1MB to 8MB. We also vary the duration between two HTTP requests (called the pause time) from 500ms to 8000ms.



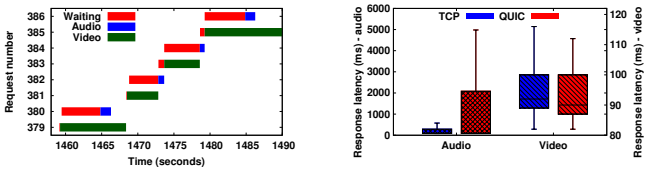
(a) Throughput (b) Response latency ($p < 0.05$ for all the instances)

Fig. 8. Performance of TCP and QUIC for two parallel connections

Fig. 8 shows the observations from this experiment. The differences in throughput between QUIC and TCP is not significant; however, the response latency with QUIC is significantly higher than TCP when two parallel application streams generate HTTP requests. Further, this difference in the response latency is more prominent when the pause time is less indicating the frequency of the HTTP requests is high. This observation is very synonymous to what we observed in Fig. 7. To find out the reason for such a behavior, we see that the problem is inherited from the behavior of the socket buffers used to interface between the user-space and the kernel-space. As TCP creates two separate sockets for the two HTTP streams, each of the sockets maintains its own socket buffer. Therefore, the HTTP responses from the two streams do not interfere with each other. On the other hand, QUIC multiplexes both the streams and uses a single UDP socket having a single socket buffer. Therefore, the HTTP responses from both the streams interfere, and higher response rate at one stream affects the queuing delay for the response at the other stream.

When DASH uses two separate HTTP streams for video and audio downloads, the stream corresponding to the video downloads has a higher data generation rate compared to the stream corresponding to the audio download. This is

because the amount of video data to be downloaded is much higher compared to the amount of the audio data to be downloaded, for a fixed playback time. For DASH/TCP, the audio and the video streams use separate socket buffers, having independent queuing delay depending on their data generation rate. However, for QUIC, both the streams get multiplexed. For every playback segment, the client generates one HTTP request for the video segment and another HTTP request for the audio segment. As the video segment request is sent first, the UDP socket buffer gets filled up with the majority of the video segment data. Consequently, the data for the audio segment needs to wait until that video data gets freed up from the socket buffer. Fig. 9a shows an example instance of video download using QUIC, where we see that video data is served almost immediately after the request is received at the server. However, the audio data has to wait in the queue (the red timeline) before it gets served. This problem is not there in TCP as the fairness property of TCP flow and congestion control serves both the socket buffers in a fair way. So, the audio data does not experience this high response latency. Fig. 9b shows the distributions of the response latency for the audio and video streams separately. We see that the distributions of the response latency for the video streams are similar for QUIC and TCP. However, the audio streams at QUIC experiences a much higher latency compared to TCP.



(a) Video/Audio downloads over QUIC

(b) Response latency for audio and video

Fig. 9. TCP and QUIC response latency during video and audio downloads over separate streams

C. Summary

We observe that during parallel downloads of audio and video streams, QUIC multiplexing affects the response latency of the audio streams. This observation also tallies with the observation made in [15] which states that multiplexing objects from parallel streams affect the latency of individual objects. This latency is inevitable because of multiplexing multiple objects over a single UDP queue, whereas the final video playback depends on successful download of both the video and the audio segments. Interestingly, the throughput calculation used in DASH does not consider this response latency, resulting in a mismatch between the expected throughput and the actual download time of the segments after a request is sent. There might not be a direct solution to this problem as DASH is unaware about such behaviors of the underlying data transmission protocol, whereas QUIC is unaware about the dependency between the audio and the video streams for successful streaming of the video. A further analysis and protocol enhancement is required in this space for making ABRs work perfectly on top of QUIC.

VI. Conclusion

This letter gives an analysis of the recent advanced ABR techniques over the QUIC as the end-to-end transport protocol. We observed that all the ABR techniques are sensitive to sudden increase or drop in the client-perceived link bandwidth, and therefore are more compatible with TCP rather than QUIC. The QUIC multiplexing of audio and video streams over a single UDP socket results in additional response latency for the audio segments, which are not captured during the calculation of channel throughput. As a consequence, the ABR algorithms take incorrect decisions during selecting the bitrates based on the calculated throughput over a QUIC connection. The analysis discussed in this letter opens up a new direction of research on exploring ABR techniques over QUIC which is the de-factor transport protocol for Google services.

References

[1] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar *et al.*, "The QUIC transport protocol: Design and internet-scale deployment," in *Proceedings of the ACM SIGCOMM*. ACM, 2017, pp. 183–196.

[2] T. Stockhammer, "Dynamic adaptive streaming over HTTP–: standards and design principles," in *Proceedings of the ACM MMSys*. ACM, 2011, pp. 133–144.

[3] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "BOLA: Near-optimal bitrate adaptation for online videos," in *Proceedings of the 35th Annual IEEE ICC*, apr 2016, pp. 1–9.

[4] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over HTTP," in *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4. ACM, 2015, pp. 325–338.

[5] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with Pensieve," in *Proceedings of the ACM SIGCOMM*. ACM, 2017, pp. 197–210.

[6] P. Biswal and O. Gnawali, "Does QUIC make the web faster?" *Proceedings of the 2016 IEEE Globecom*, 2017.

[7] P. Megyesi, Z. Krämer, and S. Molnár, "How quick is QUIC?" in *Proceedings of the 2016 IEEE ICC*. IEEE, 2016, pp. 1–6.

[8] D. Bhat, A. Rizk, and M. Zink, "Not so QUIC: A performance study of DASH over QUIC," in *Proceedings of the 27th ACM NOSSDAV*, 2017, pp. 13–18.

[9] D. Bhat, R. Deshmukh, and M. Zink, "Improving QoE of ABR streaming sessions through QUIC retransmissions," in *Proceedings of the 2018 ACM MM*, 2018, pp. 1616–1624.

[10] T. Van, H. A. Tran, S. Souihi, and A. Mellouk, "Empirical study for dynamic adaptive video streaming service based on google transport QUIC protocol," in *Proceedings of the 43rd IEEE LCN*, 2018, pp. 343–350.

[11] R. Netravali, A. Sivaraman, S. Das, A. Goyal, K. Winstein, J. Mickens, and H. Balakrishnan, "Mahimahi: Accurate record-and-replay for HTTP," in *Proceedings of the USENIX ATC*, 2015, pp. 417–429.

[12] F. C. Commission, "Raw Data - Measuring Broadband America - Eighth Report," https://www.fcc.gov/reports-research/reports/measuring-broadband-america/raw-data-measuring-broadband-america-eighth, 2018, [Online; accessed 29-March-2019].

[13] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Commute path bandwidth traces from 3G networks: Analysis and applications," in *Proceedings of the ACM MMSys*, 2013, pp. 114–118.

[14] N. Nachar *et al.*, "The Mann-Whitney U: A test for assessing whether two independent samples come from the same distribution," *TQMP*, pp. 13–20, 2008.

[15] A. M. Kakhki, S. Jero, D. Choffnes, C. Nita-Rotaru, and A. Mislove, "Taking a long look at QUIC: an approach for rigorous evaluation of rapidly evolving transport protocols," *Communications of the ACM*, vol. 62, no. 7, pp. 86–94, 2019.