Empirical study for Dynamic Adaptive Video Streaming Service based on Google Transport QUIC protocol

Van TONG*, Hai Anh TRAN†, Sami SOUIHI* and Abdelhamid MELLOUK*
*LISSI-TincNET Research Team
University of Paris-Est Creteil (UPEC), France
Email: van.tong@tincnet.fr, sami.souihi@u-pec.fr, mellouk@u-pec.fr
†Bach Khoa Cybersecurity Centre
Ha Noi University of Science and Technology, Hanoi, Vietnam
Email: anhth@soict.hust.edu.vn

***Abstract*—Quick UDP Internet Connections (QUIC) is a new transport protocol developed by Google in 2012. QUIC is considered as a combination of TCP, TLS and HTTP on the top of UDP with some advantages such as reducing connection establishment time, improving congestion control, multiplexing without head of line blocking and connection migration. In video streaming, Dynamic Adaptive Streaming over HTTP (DASH) is tied with TCP in many years, but the video streaming using HTTP on TCP has some disadvantages in term of head of line blocking, connection migration, etc. The emergence of QUIC resolves these drawbacks and provides some solutions to reduce the latency and improve the quality of network service with respect to QoE. Therefore, in this paper, we investigate and evaluate the performance of QUIC and traditional transport protocols in the context of video streaming using DASH services. Some QUIC parameters such as maximum congestion window, buffer size and number of emulated connections are considered to choose the most appropriate parameters for video streaming. Besides, we compare the performance of QUIC with TCP in term of some network parameters and some DASH parameters. The experimental results showed that the performance of QUIC with 2 emulated connections is not as good as TCP. When the number of emulated connections is set to 6, the number of changes in quality level and stalling events are lower than the figure for TCP. Consequently, the quality level of QUIC with 6 emulated connections is better than TCP. Moreover, the QoE score of QUIC with 6 emulated connections is higher than the figure for QUIC with 2 emulated connections and TCP.**

I. Introduction

Video streaming has become the most popular Internet services with huge number of service provider such as Youtube, Netflix, video streaming on Facebook, etc. Therefore, development of the adaptive streaming schemes plays an important role in many Internet services. The purpose of video streaming providers is to satisfy their end-users in improving the Quality of Experience (QoE) at the user side. In order to improve the QoE, service providers investigate Key Performance Indicators (KPIs) for a video like the buffer level, stall time, latency, screen resolution, etc.

Quality of Service (QoS) is the performance measurement of a network service. QoE has emerged from QoS that objectively measure service parameters including packet loss, bitrate, latency, throughput, etc. Reflecting another aspect of user satisfaction, QoE is a subjective measure of the user's perception of the quality of the service provided by the end-user feedback.

There are lots of research works on QoE in the context of video streaming, but they concentrate on traditional network protocols [1] [2]. The privacy in data transfer is a significant issue, so many organizations concentrate on building the Internet Protocol with encrypted network traffic. Recently, Google has developed a new transport layer network protocol, QUIC (Quick UDP Internet Connections) [3]–[6] which is implemented on the top of UDP (User Datagram Protocol). This protocol uses the advantages of traditional TCP (Transmission Control Protocol) and MTCP (Multipath Transmission Control Protocol) standard. There are five main features of QUIC including reducing connection establishment time, improving congestion control, multiplexing without head of line blocking, forward error correction, and connection migration. Moreover, QUIC also provides the security protection equivalent to TLS/SSL (Transport Layer Security/Secure Sockets Layer), so the emergency of QUIC imposes lots of challenges for service providers when the payload is encrypted.

In this paper, we extract and evaluate some parameters at the end-user side in the context of video streaming with QUIC and traditional protocols. These parameters can be used for troubleshooting the network issues in the future. We use Dynamic Adaptive Streaming over HTTP (DASH) [7] for adaptive video streaming and evaluate the performance of QUIC. In this paper, our main contributions are:

- We investigate the influence of some parameters in QUIC on video streaming process so that we can select the most appropriate parameters for particular systems.
- Previous works evaluate the performance of QUIC related to connection establishment, loss recovery, file transfer, website page load time, etc, but we consider another aspect, video streaming based on QUIC and evaluate the buffer level, throughput, latency and some DASH

parameters.
- We implement a video QoE prediction model to investigate and evaluate the QoE score of video streaming using QUIC in comparison with TCP.
- We open some interesting research directions for improving video streaming based on the QoE metrics in the context of encrypted network traffic.

The remainder of the paper is as follows. Section II presents the background and some related works. Section III describes the testbed to evaluate QUIC with others. Section IV shows experimental results and discuss some related contents. The paper concludes with Section V which highlights the conclusion and future work.

II. Background and Related work

A. DASH

Dynamic Adaptive Streaming over HTTP (DASH) [8] is known as MPEG-DASH, an adaptive bitrate streaming technique that allows high quality streaming of media content over the Internet. MPEG-DASH divides the media content into small segments. Each segment contains a short duration of playback time of media content (video, live broadcast, etc.). The media content is encoded at variety of different bitrate, and each segment is encoded at many bitrate in short interval of playback time. While playing the current segment, the MPEG-DASH clients choose automatically the next segment to download and playback with the current network condition to avoid the stalling or buffering events. The adaptive bitrate (ABR) algorithm is used to select the appropriate segments. The main goal is to optimize the user's perception and provide high quality to playback with fewer buffering and stalling events.

The library dash.js [9] is an initiative of the DASH Industry Forum for building video and audio players that playback MPEG-DASH content using client-side JavaScript libraries. In order to play a video using this player on the web browser, the clients have to download the Javascript implementation combined with an HTML video element, a part of the HTML Media Source Extensions supported by Firefox, Google Chrome, Chromium and other web browsers. There is 3 kinds of adaptive bitrate algorithm in dash.js containing throughput-based algorithm [10], BOLA algorithm [11], and hybrid heuristic. The throughput-based algorithm selects the birate based on the recent throughput while BOLA algorithm chooses the bitrate based on the buffer level. The hybrid heuristic switches between BOLA and throughput-based algorithm in real time. Dash.js uses the hybrid heuristic because this heuristic combines advantages of two above algorithms.

B. Related work

In this section, we investigate some current related work in the area of QUIC investigation. Besides, we describe the motivation to consider the video streaming based on QUIC.

Arash et al. [12] presented an alternative technique for evaluating QUIC and some previous transport protocols (TCP). They emulated the testbed to evaluate the throughput, congestion window sizes, round trip time for transferring data and loading web pages between QUIC and TCP. However, the authors only focus on the performance of QUIC with TCP in the context of data transfer.

Gaetano et al. [4] investigated the dynamics of the CUBIC congestion control algorithm between QUIC and TCP. Moreover, they measured the web page load time when HTTP/1.1, SPDY, and QUIC are employed. Gaetano considered some metrics containing goodput, channel utilization, loss ratio and page load time. The drawback of this proposal is that the testbed is a little bit simple, only 2 computers. Besides, this proposal only concentrates on the web page load time.

Megyesi et al. [13] presented a comprehensive study about the performance of QUIC, SPDY, and HTTP and evaluated the page load time in different bandwidth, packet loss and RTT. However, the testbed is simple which contains one client and one server. Moreover, the page load time is not sufficient to evaluate the performance of QUIC with other protocols.

In an M.S. thesis, Das [14] evaluated the QUIC performance based on mahimahi [15] to replay 500 real webpages over emulated network conditions. In this work, the author estimated the page load time in different bandwidth and RTT. However, this work only concentrates on page load time.

Sarah et al. [5] designed a testbed to investigate and evaluate the performance of QUIC in term of page load time against different delay, loss, etc. Besides, they also compared the page load time of QUIC with HTTP/11.1 and HTTP/2 in the context of using 4G and public ADSL links. They found that QUIC outperforms HTTP/2 on TCP/TLS in the mobile network, but the advantage of QUIC in other network conditions is not obvious. However, in this research work, the author only consider the page load time, so this can be not sufficient to evaluate the performance of QUIC.

Srivastava et al. [16] evaluated the performance of QUIC and TCP over different packet loss, throughput, and delay. Nevertheless, the author only investigates in the context of file transfer with simple testbed.

Timmerer et al. [17] evaluated the performance of QUIC and TCP on dynamic adaptive video streaming. This work focuses on the link utilization, bitrate, and throughput in some contexts. The disadvantage of this work is that the testbed is simple with only two computers, so this can lead to the uncertain experimental results.

All of the work presented above only consider the performance of QUIC in the context of data transfer, loading the web page, etc. In this paper, we investigate and evaluate QUIC in the context of video streaming using QUIC. First, we investigate the influence of maximum congestion window, buffer size and number of emulated connections on video streaming based on QUIC. Second, we build the testbed to extract the buffer level, throughput, latency and some DASH parameters at the client side and compare the performance of QUIC with traditional protocols. Last but not least, we implement a video QoE prediction model to investigate and evaluate the QoE score of video streaming using QUIC in

comparison with TCP. Our testbed can be extended to adaptive video streaming in order to improve the video streaming procedure and user's perception.

III. Testbed

In this section, we describe our testbed to evaluate the performance of HTTP/1.1 on QUIC and HTTP/1.1 on TCP in term of the buffer level, throughput, latency and some DASH parameters. Besides, we also investigate the QoE metric for DASH to evaluate the user's perception in video streaming using QUIC and some traditional protocols.

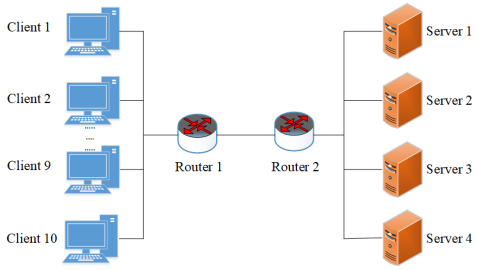

Fig. 1. The testbed setup.

To evaluate the performance of QUIC, we build the testbed with QUIC servers and QUIC clients (Fig. 1). Our experiment contains four servers, two router and ten clients with QUIC-enabled chromium browser [18]. Moreover, we use netem [19] to emulate different network conditions. Our clients consist of desktops (Ubuntu 14.04, 2GB memory, Intel Core i3 2.9GHz). The servers include desktops (Ubuntu 16.04, 8GB memory, Intel Core i5 2.5GHz) and use the quic-go libraries to implement the necessary protocol features. Library quic-go is an open-source implementation of QUIC wrote in Go language [20]. The servers also run Apache to support HTTP/1.1 on TCP.

In order to stream video from QUIC servers to QUIC clients, we use dash.js, an implementation of the MPEG-DASH standard [9]. Dash.js is an open-source which also supports the adaptive video streaming and provides some API for parameter extraction. To implement the dash.js and quic-go, we follow some instructions from their documents in [9] [18]. Besides, all experiments is implemented in the area of a university. Moreover, we also investigate and evaluate the QoE score in the video streaming using QUIC and TCP. The method for QoE prediction is re-implemented in another paper [21] using some parameters extracted from the testbed (throughput and latency).

In this paper, all experiments use the same source video, over three minutes, and we implement 3 scenarios. First, we investigate the affection of some QUIC parameters on video streaming including maximum congestion window, buffer size and the number of emulated connections. The goal of this experiment is to choose the most appropriate value in video streaming. Second, we evaluate the performance of QUIC with TCP in term of buffer level, throughput, latency and some DASH parameters. Third, we also implement a video QoE prediction model to evaluate and compare the QoE score of video streaming using QUIC and TCP.

IV. Experimental results

In this section, we conduct extensive measurements and analysis to evaluate the performance of QUIC in 3 scenarios. In the first scenario, some parameters of QUIC are changed to investigate the influence of these parameters on video streaming and select the most appropriate parameters. In the second scenario, we compare the video streaming in 3 categories consisting of HTTP-QUIC-2 (HTTP/1.1 to QUIC with 2 emulated connections), HTTP-QUIC-6 (HTTP/1.1 to QUIC with 6 number of emulated connections) and HTTP-TCP (HTTP/1.1 to TCP) in term of the buffer level, throughput, latency over the time, and some DASH parameters over the different bandwidth. In the third scenario, we evaluate and compare the user perceive in the video streaming using QUIC and TCP.

A. Evaluation metrics

To compare the performance of HTTP on QUIC with HTTP on TCP, we use some parameters that play an important role in video streaming and assessing user's perception. The detail of the metrics is as in following.

- Buffer level: Buffer level is the time when a frame is in the buffer until this frame is displayed in the web browser. When buffer level increases, the number of frames in the buffer will go up that lead to avoid the buffering events and improve the user's perception.
- Latency: The latency is the delay from input into a system to the desired outcome. In other words, the latency is the time when a frame is sent from the server until this frame is displayed on the client side. The low latency leads to the good perception of users.
- Throughput: Throughput is the amount of data during a period that a flow can send or receive. In the adaptive streaming video, the high throughput results in the low latency and better user's perception.
- Quality level: Quality level describes the quality of all displayed video segments. The higher quality level, the higher perception of users.
- Number of changes in quality level: This parameter describes the number of quality level switched in adaptive video streaming. The low throughput leads to the high number of change in quality level, and the bad influence on end-users.
- Number of stalling events: The total number of stalling events is the number of video playback disrupted. The higher number of stalling events, the lower user's perceptions.

To calculate these parameters, we use the client log provided by DASH player and some application programming interface (API) [22] supported by DASH player, except for quality level. In DASH, each segment of video content is encoded to multiple bitrates. When a segment is displayed, MPEG-DASH clients continue to request next segments from the server.

The adaptive bitrate algorithm chooses the next segments with appropriate bitrate based on the buffer level, throughput, latency and other parameters. The bitrate level will decide the quality level, and adapt to the network conditions in order to improve the user's perception. Each birate level corresponds to particular quality level.

Besides, *protocol overhead* is investigated. Protocol overhead is defined as equation 1.

$$\text{protocol overhead} = 1 - \frac{\text{media bytes}}{\text{bytes on the wire}} \quad (1)$$

Bytes on the wire is the total packet size, and media bytes is the payload of a packet. The overhead of QUIC and TCP are described as in Table I. In TCP, the maximum Ethernet payload and total packet size are limited to 1500 and 1514, respectively. In QUIC, packet size (including QUIC header) and total packet size are limited to 1350 and 1392 bytes, respectively.

TABLE I
PROTOCOL OVERHEAD OF QUIC AND TCP

	TCP	**QUIC**
QUIC header		22
UDP/TCP header	32	8
IP and Ethernet header	34	34
Total header	66	64
Maximum packet size	1514	1392
Overhead	4.35%	4.6%

B. QUIC parameter selection

In this section, we analyze the difference of the number of emulated connections, buffer size and maximum congestion window in video streaming. In quic-go [20], the default buffer size is set to 1MB, and the default maximum congestion window is 1000. Besides, the default number of emulated connections is set to 2.

1) Buffer size: QUIC employs connection-level flow control and the stream-level flow control [3]. The connection-level flow control limits the aggregate buffer that QUIC servers can use at the QUIC clients across all streams while stream-level flow control limits the buffer that QUIC servers can use on any given stream.

In this section, we change the buffer size in stream-level flow control (buffer size $\in$ [0.5, 1, 1.5, 2] (MB)) and evaluate buffer level, throughput and latency of video streaming using QUIC (Table II). To achieve maximum throughput, it is critical to use optimal buffer size. If the buffer size is to small, the congestion window will not open up fully, and the QUIC server will be throttled. Otherwise, the QUIC server can overrun the QUIC client which causes the QUIC client to drop packets.

As usual, the buffer size in stream-level flow control is lower than the figure for connection-level flow control, so this allows multiplex connections in video streaming. However, the default buffer size in connection-level flow control is 1.5 MB, so QUIC server cannot support multiplex connections when the buffer size in stream-level flow control is equal or bigger than 1.5MB. As a result, the average buffer level decreases slightly from 28.82 to 26.31 seconds whereas the average throughput reduces significantly from approximately 32,000 to over 30,000 kbps.

TABLE II
INFLUENCE OF BUFFER SIZE IN STREAM-LEVEL FLOW CONTROL ON VIDEO STREAMING

Buffer size (MB)	**0.5**	**1**	**1.5**	**2**
Average buffer level (s)	28.82	28.81	28.70	26.31
Average throughput (kbps)	31,952	31,790	31,541	30,328
Average latency (ms)	11.42	12.03	12.47	12.42

The latency is directly related to the throughput and buffer level. The difference in throughput is clear while the buffer level decreases slightly. Therefore, there is an upward trend in the average latency from 11.42 to 12.47 milliseconds.

2) Congestion window size: The congestion window is the value that limits the amount of data can send into the network before receiving the responses. In quic-go, the default max congestion window size is set to 1000. We change the maximum congestion window from 500 to 2500 and investigate the difference of buffer level, throughput, and latency in video streaming (Table III).

TABLE III
INFLUENCE OF NUMBER OF MAXIMUM CONGESTION WINDOW ON VIDEO STREAMING

Congestion window	**500**	**1000**	**1500**	**2000**	**2500**
Average buffer level (s)	28.16	28.93	28.76	28.90	28.91
Average throughput (kbps)	28,229	29,661	30,296	30,520	30,188
Average latency (ms)	13.35	13.04	12.66	11.70	12.52

The average buffer level fluctuates with an upward trend when the maximum congestion window increases from 500 to 2,500. At some moment of the beginning of the video streaming produce, the throughput decreases slightly which leads to the corresponding of reduction of buffer level. As a result, the average buffer level at the maximum congestion window of 1500 is not as high as others and the fluctuation in the average buffer level. During the first 10 seconds, the buffer level increases remarkably to over 29 seconds. The average buffer level is approximately 28.93 seconds at the maximum congestion window of 1000, and then the figure decrease to 28.76 seconds at the maximum congestion window of 1,500. The reason for it is that the buffer level reduces suddenly at some moment in video streaming procedure, so the average buffer level is lower at the maximum congestion window of 1500.

Concerning the throughput, there is a significant increase

in the average throughput from over 28,000 to approximately 30,500 kbps. The maximum congestion window limits the amount of data which can send into the network, so the average throughput goes up when this value increases. When maximum congestion window is set to 2500, the loss is detected, and this leads to the reduction of average throughput.

As in the Table III, the average latency has a downward trend when the maximum congestion window increases. The average throughput goes up against the maximum congestion window while the average buffer level is almost unchanged. This will lead to the decrease of average latency. Average latency is 13.35 milliseconds at the maximum congestion window of 500 whereas the figure is 11.7 milliseconds at the maximum congestion window of 2,000. When the maximum average latency increases from 2,000 to 2,500, the average throughput falls to over 30,000 kbps. Consequently, the average latency climbs to 12.52 milliseconds.

In five value we investigated, we found that 2000 is the best maximum congestion window. This value is also used by chromium [18], but the default maximum congestion window is 1000 in quic-go [20].

3) Number of emulated connections: In QUIC, there is a constant variable for the number of emulated connections in a congestion control module. In video streaming using DASH, the default number of emulated connections is set to 2. In other words, one QUIC connection is approximately empowered as aggressive as two TCP connections. The number of emulated connections is adjusted in order to force QUIC to transmit data robustly. We stream video with a different number of emulated connections (number of emulated connections $\in$ [2, 4, 6, 8, 10]) and calculate the average buffer level, throughput, and latency of video streaming using QUIC (Table IV).

TABLE IV
INFLUENCE OF NUMBER OF EMULATED CONNECTIONS ON VIDEO STREAMING

Number of emulated connections	**2**	**4**	**6**	**8**	**10**
Average buffer level (s)	28.8	28.9	28.4	28.9	28.93
Average throughput (kbps)	25,090	32,708	32,568	33,117	31,288
Average latency (ms)	16.03	13.55	12.99	13.05	13.22

DASH is an adaptive bitrate streaming technique that enables high-quality streaming of media content over the Internet. Moreover, the buffer level is limited to 30 seconds in dash.js. Therefore, the average buffer level nearly remains unchanged with approximately 28.9 seconds while the number of emulated connections increases from 2 to 10. During the period of video streaming, the buffer level goes up significantly from 0 to over 29 seconds in the first ten seconds and then fluctuates between 29 and 30 seconds.

Regarding the throughput, there is an upward trend in the average throughput when the number of emulated connections increases between 2 and 10. Every connection, QUIC begins in the stage of the slow start and finishes until loss [6]. When acknowledge is processed, the number of congestion window size increases in the stage of the slow start. Therefore, the increasing of the number of emulated connections results in the upward trend in congestion window. As a result, the average throughput significantly climbs from approximately 25,000 to over 33,000 kbps. A noticeable feature from the Table IV is that the throughput decreases to over 31,000 kbps with 10 emulated connections. When the number of emulated connection is set to 10, the congestion window increases. However, the default max congestion window is not high enough, so the loss will exist. QUIC will finish the slow start to avoid the loss and congestion. When a loss is detected, TCP cubic reduces the congestion window and set another slow start threshold to the new congestion window.

As for the average latency, the figure for 6 emulated connections is the lowest with 12.99 milliseconds. When the number of emulated connections increases from 2 to 6, the throughput climbs significantly to over 32,000 kbps. This is the reason for the reduction of latency between 2 and 6 emulated connections. However, the number of emulated connections is larger than 6, the congestion window and buffer size reach the maximal value. After that, the congestion window size and buffer size cannot be increased whereas the data increased robustly. Therefore, the average latency goes up to 13.22 milliseconds with 10 emulated connections.

When the number of emulated connections increases from 2 to 10, we found that 6 is the appropriate value. With 6 emulated connections, the average throughput is the highest, and the average latency is the lowest in 5 categories.

C. Comparison between QUIC and TCP

In above section, we investigate the performance of QUIC with the different value for the number of emulated connections. We found that 6 is the appropriate value for the number of emulated connections, so we compare the performance of TCP with QUIC-2 (QUIC with 2 emulated connections), QUIC-6 (QUIC with 6 emulated connections) and TCP in the video streaming.

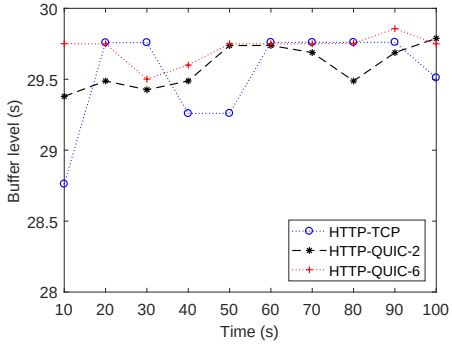

Fig. 2. The comparison of buffer level (s) over the time between HTTP-TCP, HTTP-QUIC-2 and HTTP-QUIC-6.

In Figure 2, there is the significant difference between HTTP-TCP, HTTP-QUIC-2, and HTTP-QUIC-6 in the first 60 seconds. In the period of 20-30 seconds, the buffer level of HTTP-TCP is the highest with approximately 29.7 seconds, and then this figure reduces to 29.26 seconds at the time of 50 seconds. In general, the buffer level of HTTP-QUIC-6 is slightly higher than the figure for HTTP-QUIC-2 in the first 50 seconds. During the last 40 seconds, the buffer level of HTTP-TCP, HTTP-QUIC-2, and HTTP-QUIC-6 are almost same with about 29.8 seconds.

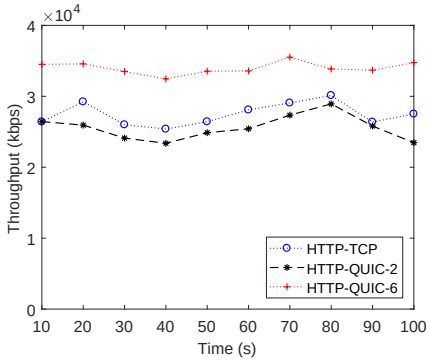

Fig. 3. The comparison of throughput (kbps) over the time between HTTP-TCP, HTTP-QUIC-2 and HTTP-QUIC-6.

In Table I, the maximum packet size of TCP is slightly higher than the figure for QUIC with 1514 and 1392 bytes, respectively. Besides, the overhead of QUIC is 4.6 % while the figure for TCP is 4.35 %. Therefore, the throughput of HTTP-QUIC-2 is slightly lower than HTTP-TCP in the duration of 100 seconds. When the number of emulated connection is set to 6, the throughput of HTTP-QUIC-6 is the highest in 3 categories because one QUIC connection is approximately empowered as aggressive as 6 TCP connections. The throughput of HTTP-QUIC-6 is over 32,000 kbps in the duration of 100 seconds. The detail is described as in Figure 3. The throughput of HTTP-QUIC-2 fluctuates during the time of video streaming. Firstly, QUIC servers will stream video robustly to QUIC clients until reaching a peak of buffer level in Dash.js and buffer size in stream-level flow control. After that, QUIC reduces the intensity of video streaming to avoid the congestion and loss. Then, when buffer level and buffer size are lower than a threshold, QUIC servers continue to stream video robustly. As a result, in HTTP-QUIC-2, the throughput decreases to over 26,000 kbps at the time of 40 seconds, and then continue to reach a peak at approximately 29,000 kbps at the time of 80 seconds. Then the throughput reduces by 600 kbps at the time of 100 seconds. It is similar to HTTP-QUIC-6.

In the context of video streaming, latency is one of the essential factors. Figure 4 shows the difference of latency between QUIC and traditional protocols. As described in Fig. 3, the throughput of HTTP-QUIC-2 is slightly lower than the curve for HTTP-TCP. Therefore, the latency of HTTP-QUIC-2 is higher than the figure for HTTP-TCP. When the number of emulated connections increases to 6, the throughput of HTTP-

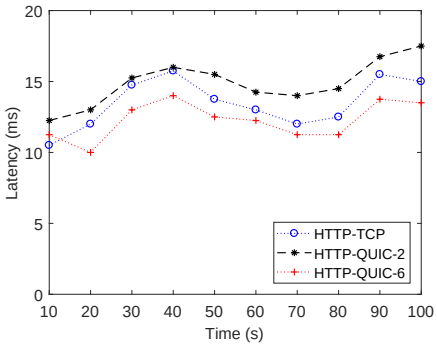

Fig. 4. The comparison of latency (ms) over the time between HTTP-TCP, HTTP-QUIC-2 and HTTP-QUIC-6.

QUIC-6 is larger than the curve for HTTP-TCP. Consequently, the latency of HTTP-QUIC-6 is the lowest in three categories during the period of 100 seconds.

In Figure 3, the curve of 3 categories reduce during the first 40 seconds, and then significantly increase between the time of 40 and 80 seconds. Therefore, the latency of 3 categories reach a peak at the time of 40 seconds and then hit the bottom at the time of 80 seconds. The latency of HTTP-QUIC-6 is 11.25 milliseconds at the time of 10 seconds and hit a bottom with 10 milliseconds at the time of 20 seconds. The figure of HTTP-QUIC-6 increases between the time of 20 and 40 seconds and then decreases slightly to 11.25 milliseconds at the time of 80 seconds. At the last 10 seconds, the latency of HTTP-QUIC-6 remains unchanged with the approximately 13.5 milliseconds.

We not only consider the buffer level, throughput and latency, but also we investigate some DASH parameters such as the number of changes in quality level, number of stalling events and quality level against bandwidth. The detail is described as in Table V. To evaluate and illustrate significantly difference of the DASH parameters on the video streaming, we use netem to change the bandwidth of the network (bandwidth $\in$ [3,000, 5,000, 7,000, 10,000](kbps)).

The first DASH parameter is the number of stalling events. The stalling is the interruption of video playback due to the empty buffer. Stalling exists until the buffer gets enough segments to resume the video playback. The lower bandwidth, the higher number of stalling events. The throughput and latency of HTTP-QUIC-6 are the better than the others, so the number of stalling events in HTTP-QUIC-6 is the lowest in three categories. When bandwidth is set to 3000 kbps, the number of stalling events of HTTP-TCP, HTTP-QUIC-2 and HTTP-QUIC 6 are 2,1,0, respectively. In the first 10 seconds of video streaming, the low bandwidth results in the slight increase of the buffer level. When the adaptive bitrate chooses the higher quality level, the buffer level decreases significantly to 0 that leads to 2 times of the stalling events of HTTP-TCP during the first 10 seconds. It is similar to 1 time of stalling events of HTTP-QUIC-2 at the time of 110 seconds. A noticeable feature of the table is that there is stalling events

TABLE V
COMPARISON OF HTTP-TCP, HTTP-QUIC-2 AND HTTP-QUIC-6 IN TERM OF NUMBER OF CHANGES IN QUALITY LEVEL, NUMBER OF STALLING EVENTS AND QUALITY LEVEL

Bandwidth (kbps)	Number of changes in quality			Number of stalling events			Average quality level		
	HTTP-TCP	HTTP-QUIC-2	HTTP-QUIC-6	HTTP-TCP	HTTP-QUIC-2	HTTP-QUIC-6	HTTP-TCP	HTTP-QUIC-2	HTTP-QUIC-6
3,000	15	25	19	2	1	0	2.85	2.58	3.89
5,000	7	13	16	0	0	0	4.25	3.37	4.3
7,000	7	12	7	0	0	0	4.86	4.84	5.34
10,000	8	12	4	2	1	1	5.3	5.13	5.5

with the bandwidth of 10,000 kbps. These stalling events are in the first 10 seconds of video streaming duration. Although the bandwidth is set to 10,000 kbps, the buffer level is very small (less than 3) in the first 10 seconds. The adaptive bitrate tries to request the next segment with high-quality level, so this leads to the reduction of buffer level which results in the stalling events in three categories. After that, the buffer level increases, so the stalling events do not exist. Besides, the number of stalling events of HTTP-TCP is higher than the figure for HTTP-QUIC-2 and HTTP-QUIC-6 due to the slow increase of buffer level in HTTP-TCP during the first 10 seconds.

The second DASH parameter is the number of changes in the quality level. When the bandwidth goes up from 3,000 to 10,000 kbps, the number of changes in the quality level decreases remarkably in three categories. The number of changes in the quality level of HTTP-QUIC-2 is not good. As explained in Tab. I, the overhead of QUIC with 2 emulated connections is bigger than the figure for TCP, so the throughput of HTTP-QUIC-2 is not as good as the others. Therefore, the adaptive bitrate algorithm in DASH selects the segments with appropriate quality to avoid buffering and stalling events. Consequently, the number of change in the quality level of HTTP-QUIC-2 is not good. Moreover, the figure for HTTP-QUIC-6 is the best in three categories.

The last QoE parameter is average quality level. This parameter describes the quality of video displayed. The number of stalling events is one of the essential factor influencing on the average quality level. The high number of stalling events, the lower average quality level. In Table V, the number of stalling events and the number of changes in the quality level of HTTP-QUIC-6 are lower than others. Besides, the throughput of HTTP-QUIC-6 is larger than the others, so the average quality level of HTTP-QUIC-6 is the highest in three categories. Moreover, the average quality level of three categories increases significantly when the bandwidth climbs from 3,000 to 10,000 kbps. In the bandwidth of 10,000 kbps, there is some stalling events, but the global quality level is high. Consequently, the average quality level is not decreased in this bandwidth.

D. Video QoE prediction

In this paper, we not only define the appropriate parameters for video streaming using QUIC, compare the performance of QUIC and TCP related to buffer level, latency, throughput and some DASH parameters, but also we estimate and evaluate the video QoE in the context of video streaming based on QUIC and TCP. We collect 100 samples (throughput and latency) using our testbed (in section III). Based on the method proposed in [21], we re-implement the video QoE prediction model with some alternatives to estimate the video QoE of 100 samples.

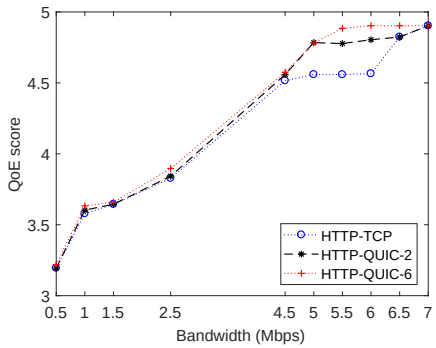

Fig. 5. The comparison of QoE score over bandwidth between HTTP-TCP, HTTP-QUIC-2 and HTTP-QUIC-6.

Figure 5 indicates the comparison of QoE score between HTTP-TCP, HTTP-QUIC-2 and HTTP-QUIC-6. There is an upward trend in the QoE score of three categories from 3.2 to 4.9. The QoE score of HTTP-QUIC-2 and HTTP-TCP are nearly equal in 10 value of bandwidth. In Table V, the stalling events frequently exist and the quality level is not good enough when the bandwidth fluctuate between small value. Therefore, the QoE score fluctuates with approximately 3.7. Besides, the QoE score of HTTP-QUIC-6 is slightly higher than the curve for HTTP-QUIC-2 and HTTP-TCP during the first 4 bandwidth. When the bandwidth is bigger than 4.5Mbps, the number of stalling events is not substantial, and the quality level is better. Consequently, there is a significant larger in the QoE score of HTTP-QUIC-6 than HTTP-QUIC-2 with approximately 4.9 and 4.78, respectively. Especially, the QoE score of HTTP-QUIC-6 is over 4.9 during the last 3 bandwidth. As in Fig. 5, the user's perception of the video streaming using QUIC with 6 emulated connections are better than the curve for TCP.

V. CONCLUSION

In this paper, we presented the empirical study for adaptive video streaming using QUIC protocol. We investigated some QUIC parameters and the influence of these parameters on video streaming. The reason for these experiments is that we

want to choose the most appropriate parameters for video streaming. 2 is the default value for the number of emulated connections, but we found that 6 is the suitable value due to the high throughput and low latency. As for the buffer size in stream-level flow control and maximum congestion window, the better values are 0.5 MB and 2000, respectively. Moreover, we compared QUIC protocol with TCP protocol in term of buffer level, throughput, latency, and some DASH parameters. The throughput of HTTP-QUIC-2 is lower than the figure for HTTP-TCP, so the latency is larger than the curve for HTTP-TCP. When the number of emulated connection is increased to 6, the throughput of HTTP-QUIC-6 is larger than the curve for HTTP-TCP, so this leads to the reduction of latency. Besides, we investigate the QoE score of video streaming based on QUIC and TCP. The results obtained proved that the video streaming using QUIC with 6 emulated connections provides the better user's perception than TCP.

We believe that the results of this article may open up other interesting research directions. First, QUIC need to be investigated and evaluated in different environments including the wired network, wireless network and cellular network. Second, this paper can be extended to improve the service quality of network systems in the context of encrypted network traffic. This is achieved by analyzing the interaction between the user and the web browser with the parameters extracted in video streaming. The user feedback can be used for creating an adaptive loop for improving the service quality of the network systems. Third, there are lots of adaptive bitrate algorithms, so this paper can be extended to investigate and compare the performance of QUIC with TCP in the context of using different adaptive bitrate algorithms to select the most appropriate algorithm for particular systems. Last but not least, this paper can be extended to build the architecture CDN which make use of QoE result unlikely the traditional architectures that make use of QoS metrics, in the context of encrypted network traffic. The architecture CDN can help to choose the most appropriate CDN server for improving the quality of video and user's perception.

REFERENCES

[1] N. Eswara, K. Manasa, A. Kommineni, S. Chakraborty, H. P. Sethuram, K. Kuchi, A. Kumar, and S. S. Channappayya, "A continuous qoe evaluation framework for video streaming over http," *IEEE Transactions on Circuits and Systems for Video Technology*, 2017.

[2] M. F. M. Hossain, M. Sarkar, and S. H. Ahmed, "Quality of experience for video streaming: A contemporary survey," in *Wireless Communications and Mobile Computing Conference (IWCMC), 2017 13th International*. IEEE, 2017, pp. 80–84.

[3] 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 Conference of the ACM Special Interest Group on Data Communication*. ACM, 2017, pp. 183–196.

[4] G. Carlucci, L. De Cicco, and S. Mascolo, "Http over udp: an experimental investigation of quic," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. ACM, 2015, pp. 609–614.

[5] S. Cook, B. Mathieu, P. Truong, and I. Hamchaoui, "Quic: Better for what and for whom?" in *Communications (ICC), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1–6.

[6] I. Swett and J. Iyengar, "Quic loss recovery and congestion control," *Internet-Draft, Intended status: Informational, Internet Engineering Task Force*, 2015.

[7] M. P. E. Group, "Iso/iec 23009, mpeg-dash." Available online at http://mpeg.chiariglione.org/standards/mpeg-dash (Jan, 2018).

[8] T. Stoekhammer, "Dynamic adaptive streaming over http-design principles and standards," in *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, vol. 2014. New York, USA: ACM, 2011, pp. 2–4.

[9] "Dash.js." Available online at https://github.com/Dash-Industry-Forum/dash.js (Jan 2018).

[10] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive," *IEEE/ACM Transactions on Networking (TON)*, vol. 22, no. 1, pp. 326–340, 2014.

[11] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "Bola: Near-optimal bitrate adaptation for online videos," in *INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications, IEEE*. IEEE, 2016, pp. 1–9.

[12] A. M. Kakhki, S. Jero, D. Choffnes, C. Nita-Rotaru, and A. Mislove, "Taking a long look at quic," 2017.

[13] P. Megyesi, Z. Krämer, and S. Molnár, "How quick is quic?" in *Communications (ICC), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1–6.

[14] S. R. Das, "Evaluation of quic on web page performance," *Master's thesis, Massachusetts Institute of Technology*, 2014.

[15] R. Netravali, A. Sivaraman, S. Das, A. Goyal, K. Winstein, J. Mickens, and H. Balakrishnan, "Mahimahi: Accurate record-and-replay for http." in *USENIX Annual Technical Conference*, 2015, pp. 417–429.

[16] A. Srivastava, "Performance analysis of quic protocol under network congestion," Ph.D. dissertation, WORCESTER POLYTECHNIC INSTITUTE, 2017.

[17] C. Timmerer and A. Bertoni, "Advanced transport options for the dynamic adaptive streaming over http," *arXiv preprint arXiv:1606.00264*, 2016.

[18] Google, "chromium." Available online at https://www.chromium.org/quic (Jan 2018).

[19] "Network emulator." Available online at http://man7.org/linux/man-pages/man8/tc-netem.8.html (Jan 2018).

[20] "Quic-go." Available online at https://github.com/lucas-clemente/quic-go (Jan 2018).

[21] L. Amour, S. Sami, M. S. Mushtaq, S. Hoceini, and A. Mellouk, "Perceived video quality evaluation based on interactive/repulsive relation between the qoe ifs," in *Communications (ICC), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1–7.

[22] "Api of dash.js." Available online at http://cdn.dashjs.org/latest/jsdoc/module-MediaPlayer.html (Jan 2018).