Streaming 360° Videos to Head-Mounted Virtual Reality Using DASH over QUIC Transport Protocol

Shou-Cheng Yen
Department of Computer Science
National Tsing Hua University

Ching-Ling Fan
Department of Computer Science
National Tsing Hua University

Cheng-Hsin Hsu
Department of Computer Science
National Tsing Hua University

ABSTRACT

We design, implement, and evaluate a tiled DASH streaming system for 360° videos using QUIC/UDP protocol, in which multiplexed and prioritized streams are leveraged for sending urgent tiles that are about to miss their playout time. In particular, we develop a new architecture to concurrently request for regular tiled segments at a lower priority and urgent tiled segments at a higher priority as multiple streams over a single QUIC connection. Several core components, including the fixation prediction algorithm, fast tile selector, and Adaptive Bit Rate (ABR) controller are designed for this new architecture. Our trace-driven experiments reveal that: (i) DASH streaming over the QUIC protocol outperforms doing that over the HTTP/1.1 and HTTP/2 stacks and (ii) our urgent tiled segments reduce the missing ratio and increase the video quality without incurring excessive bandwidth utilization under diverse network bandwidth, user behavior, and video characteristics.

CCS CONCEPTS

• **Information systems → Multimedia streaming**; • **Networks → Transport protocols**; • **Human-centered computing → Virtual reality**.

KEYWORDS

360° Video, Video Streaming, QUIC, Virtual Reality, DASH, Design, Prototype

ACM Reference Format:
Shou-Cheng Yen, Ching-Ling Fan, and Cheng-Hsin Hsu. 2019. Streaming 360° Videos to Head-Mounted Virtual Reality Using DASH over QUIC Transport Protocol. In *24th Packet Video Workshop (PV '19), June 18, 2019, Amherst, MA, USA*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3304114.3325616

This work was partially supported by the Ministry of Science and Technology of Taiwan (#107-2221-E-007-091-MY3 and #107-2918-I-007-014).

1 INTRODUCTION

360° videos enable users to view the surrounding scenes in arbitrary orientations, which have become very popular in both the academia and industry. The popularity of Head Mounted Display (HMD) is also growing, which allows people to turn their heads to experience 360° videos instead of navigating through a mouse or other less intuitive input devices. Streaming 360° videos over the best-effort Internet to HMDs, however, is inherently difficult for two reasons:

- *High bandwidth.* Next generation Virtual reality (VR) systems require 50–200 Mbps bandwidth and 6 DoF videos require over 200 Mbps bandwidth [19, 25].
- *Low latency.* Extremely low response time (20–30 ms) to head movements must be achieved to avoid sickness [19].

In the literature, the high bandwidth requirement is coped with *tiled streaming*, which cuts each video frame into equal-sized rectangular tiles that can be *selectively* streamed to HMDs [11]. By doing so, a VR system may focus on the tiles overlapped with the current viewport (typically 67°×67°–100°×100°) to reduce the bandwidth requirement. While the tiled streaming systems [11, 17, 33] consume less bandwidth, they are vulnerable to *missing* tiles[2], which in turn results in either: (i) some *black holes* in the viewport or (ii) playout *freezes* due to waiting for the missing tiles. Such negative impacts on HMD user experience *will* drive users away from VR services.

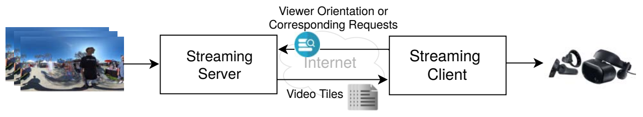

Figure 1: 360° video streaming needs high bandwidth, low latency.

The *missing-tile* issue is further *amplified* by the dominating streaming protocol: Dynamic Adaptive Streaming over HTTP (DASH) [28], which employs HTTP over TCP for video streaming. While DASH is quite suitable for *presentational* or unidirectional video streaming, it is not suitable for 360° tiled video streaming that is more interactive, because of its inherent extra delay. As illustrated in Fig. 1, the HMD orientations or sensor data determine the tiles that will fall in the viewports, and the corresponding requests *must* be sent to the streaming server in time. *Therefore, naively applying DASH for 360° video streaming may result in suboptimal streaming quality.* Based on this observation, we may switch to another extreme design and adopt Real-time Transport Protocol (RTP) for high responsiveness. Doing so however complicates the

[2]In real-time streaming, late tiles (packets) are considered as missing tiles (packets).

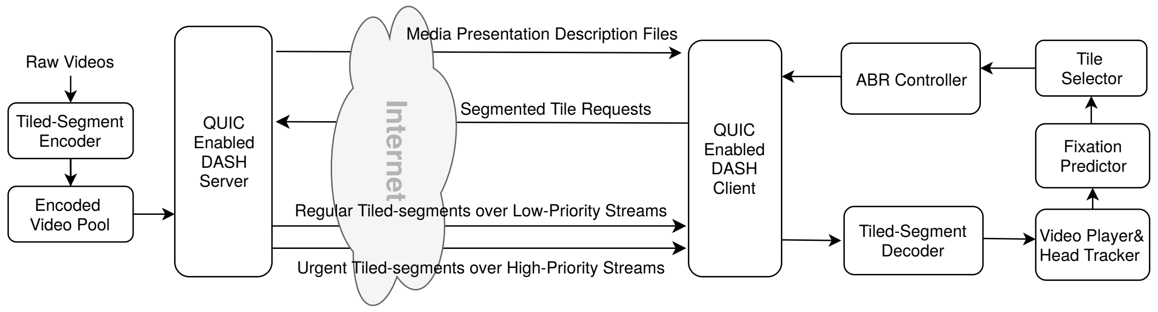

Figure 2: Architecture of the 360° video streaming using DASH over QUIC protocol.

360° video streaming systems, because of the UDP-related details, such as flow/congestion control and reliable transmission.

In this paper, we adopt the QUIC protocol [14] to optimize 360° video DASH streaming to HMDs. QUIC was initially created by Google, and has been adopted as an IETF standard. QUIC runs on UDP, and was designed to replace the HTTP/2, TLS, and TCP protocols in the HTTPS stack. The emerging QUIC has three main features: (i) secured communications, (ii) multiplexed streams with prioritized schedulers, and (iii) low latency at the time of writing. We make a critical observation: *stream multiplexing* and *low latency* are the key enablers to optimize 360° tiled video DASH streaming. This is because *when a viewer suddenly rotates his/her head, QUIC allows the system to quickly send urgent tiles at high priority to mitigate the negative impacts due to missing tiles.* This approach provides higher *interactivity* while retaining the *simplicity* of DASH streaming. Several DASH components, however, need to be optimized for capitalizing the unique QUIC features. For example, existing Adaptive Bit Rate (ABR) algorithms [13] are designed for unprioritized/sequential video streaming, which may not work well with QUIC.

We tackle the aforementioned challenges in two directions:

- First, we design and implement a QUIC-based DASH streaming system on a few open-source projects [5, 7, 24]. We then evaluate our system through real experiments driven by a public HMD viewer dataset [16]. To our best knowledge, 360° tiled video DASH streaming over QUIC has not been experimentally evaluated.
- Second, we optimize the proposed system by realizing a few key components: (i) *fixation predictor* that predicts the user viewports in the future, (ii) *tile selector* that maps (future) viewports to tiles for DASH requests, and (iii) *ABR controller* that interfaces with existing ABR algorithms to control prioritized streams.

The rest of this paper is organized as follows. Sec. 2 surveys the related work. This is followed by the system architecture given in Sec. 3. Sec. 4 describes several customized components. We evaluate our system in Sec. 5. Sec. 6 concludes the paper.

2 RELATED WORK

360° tiled video streaming over HTTP/2. HTTP/2 over TCP has been adopted by some 360° tiled streaming studies in the literature. For example, Petrangeli et al. [27] use the server push mechanism of HTTP/2 to reduce the network overhead. That is, the streaming client only sends one request for all the tiles of a single segment. Stefano et al. [26] divide each video into multiple tiles and assign different quality levels to individual tiles, based on the user fixation. There are studies [22, 30] proposing ABR algorithms to leverage two HTTP/2 features: stream priority and early termination to better handle the network fluctuations and unpredictable head movements. Similar to our work, Ben Yahia et al. [31] utilize multiplexed and prioritized streams supported by HTTP/2 to deliver urgent tiles. In particular, they call the fixation prediction algorithm twice: 1-sec and 500-ms before the playout time of each segment. They request the tiles within 500-ms playout time with higher priority. Different from our approach that employs QUIC over UDP transport, their proposal is built upon HTTP/2 over TCP. Moreover, they resort to simulations for performance analysis, while we build and evaluate a real 360° DASH video streaming system.

Video streaming over QUIC. Several studies in the literature investigate the potentials of QUIC on video streaming. For instance, Arisu and Begen [1] evaluate the performance of QUIC video streaming over wireless and cellular networks. They find that QUIC over UDP outperforms HTTP/2 over TCP, especially when the network is congested. Timmerer and Bertoni [29] also quantitatively compare the performance of video streaming over QUIC with other transport protocols. Bhat et al. [4] consider several state-of-the-art ABR algorithms to compare the video quality delivered by QUIC and HTTP/2. They find that existing ABR algorithms tend to switch to lower quality levels when QUIC is used. To address the problem, they propose retransmission over QUIC [3] to reduce the number of quality switches and increase the average bitrate. Palmer et al. [23] extend QUIC to support unreliable streams, and their proposed solution outperforms the ordinary HTTP/2 over TCP and QUIC over UDP. The aforementioned studies do no target 360° video DASH streaming, where extremely high bandwidth is required. Hayes et al. [8–10] partially solve the bandwidth problem with MultiPath TCP (MPTCP) and employ QUIC to mitigate the network congestion and fluctuation problems due to the large reordering buffers of MPTCP. Their studies, unfortunately, do not utilize the advantages of tiled streaming and always transmit the whole 360° video frames.

Different from the prior arts, our work is the first study using a real testbed implementation of 360° tiled video DASH streaming over QUIC/UDP protocol, where multiplexed and prioritized streams are leveraged for sending urgent tiles.

3 PROPOSED SYSTEM ARCHITECTURE

Fig. 2 summarizes the components in our proposed streaming system. We briefly introduce individual components below.

- **Tiled-segment encoder** contains an HEVC encoder and an MPEG DASH content generator, which first cuts each 360° video into tiles, and then compresses them into *tiled segments*. Each tiled segment is: (i) a few seconds long, (ii) independently decodable, and (iii) a basic transmission unit.
- **Encoded video pool** stores the encoded tiled segments of each video at different quality levels, which are determined by control parameters like Quantization Parameters (QPs), resolutions, or bitrates.
- **QUIC enabled DASH server** is an enhanced web server, which also supports HTTPS/QUIC. It sends prioritized tiled segments over multiplexed streams.
- **Video player & head tracker** displays 360° videos to the viewer and keeps track of the viewer orientations.
- **Fixation predictor** estimates the future fixation based on the HMD orientation and video content.
- **Tile selector** determines the tiled segments to request based on the fixation prediction results to avoid missing tiles.
- **ABR controller** decides: (i) when to request each titled segment and (ii) at which priority level (regular vs. urgent).
- **QUIC enabled DASH client** generates the regular/urgent tiled segment requests. Each stream carries only one tile request and all streams are within a single QUIC connection.
- **Tiled-segment decoder** decodes the tiled segments received from the server. It also synchronizes the tiled segments of the same segment duration.

4 COMPONENT DESIGNS

In this section, we present the design of our core components.

4.1 Fixation Predictor

Several algorithms can be used for fixation prediction, e.g., Dead-Reckoning (DR) [20] and neural-network [6, 21] based algorithms. We implement a DR algorithm based on speed and acceleration [20]. Let θ^c and ϕ^c be the yaw and pitch (in degrees) of the viewport center. We write their angular speeds at time t as $v_t^{\theta^c}$ and $v_t^{\phi^c}$; and their angular accelerations as $a_t^{\theta^c}$ and $a_t^{\phi^c}$. These values are computed as weighted moving averages, where the sampling rate is 500 ms and the weight of the latest measurement is 0.9, if not otherwise specified. The DR algorithm predicts the viewport center τ s later than the current time t as:

$$\hat{\theta}^c_{t+\tau} = \theta^c_t + v_t^{\theta^c}\tau + (1/2)a_t^{\theta^c}\tau^2; \tag{1a}$$

$$\hat{\phi}^c_{t+\tau} = \phi^c_t + v_t^{\phi^c}\tau + (1/2)a_t^{\phi^c}\tau^2. \tag{1b}$$

We note that some DR algorithms omit the acceleration terms. We empirically test both versions with the dataset [16] and find that skipping the acceleration terms results in fewer missing tiles[3]. Thus we report the results from the DR algorithm without considering acceleration.

4.2 Tile Selector

The HMD viewport can be described by the center (θ^c, ϕ^c) and the radius r_v. The tile selector finds the required tiles by: (i) creating a viewport plane that is tangent to the sphere at the viewport center, (ii) projecting the points from the viewport to the sphere, (iii) mapping the points from the sphere to the 360° video content, and (iv) identifying the overlapped tiles. The process is however time-consuming and we propose a real-time algorithm to *approximate* it. We consider the popular equirectangular projection in our discussion, while the same concepts can be applied to other projections.

The key idea is to approximate the viewport on the video content with the center as (x^c, y^c) and the width as a function of the pitch value ϕ^ϵ. The derivation of the center is straightforward: $x^c = W\frac{\theta^c+180°}{360°}$ and $y^c = H\frac{90°-\phi^c}{180°}$, where $W \times H$ is the resolution in pixel. The pitch values in the viewport are between $\phi^c \pm r_v$, and we approximate the width at $\phi^c - r_v \le \phi^\epsilon \le \phi^c + r_v$ as:

$$w(\phi^\epsilon) = \frac{2r_v\sqrt{1-\cot^2 r_v \cdot \tan^2(\phi^\epsilon - \phi^c)}}{\cos\phi^\epsilon}\frac{W}{360°}. \tag{2}$$

Having (x^c, y^c) and $w(\phi^\epsilon)$, we can compute the overlapped tile set $S(r_v)$. More specifically, we start from the center and move up/down at integer number of tiles without exceeding the viewport. At each tile row, we compute $w(\phi^\epsilon)$ to determine the width of the viewport on the video content. In rare cases where $\phi^\epsilon < -90°$ or $\phi^\epsilon > 90°$, we let $w(\phi^\epsilon) = W$. We note that while the viewport radius r_v should be no smaller than that of the HMD; it can be set slightly *larger* to accommodate the imperfect fixation prediction.

4.3 ABR Controller

Our ABR controller consists of two threads: (i) *regular tile* thread, which is event triggered and (ii) *urgent tile* thread, which is time triggered. More precisely, the regular tile thread is activated only when: (i) all previously requested regular tiled segments are received and (ii) the current buffer occupancy doesn't exceed the buffer high watermark B_s, which is a system parameter. Otherwise, the regular tile thread sleeps until both conditions are met. The urgent tile thread is triggered once every P s if the current buffer occupancy is not lower than the buffer low watermark B_l, where the *urgent window* P and B_l are system parameters.

Upon being invoked at time t, the urgent tile thread requests for the tiled segments from $S(r_v)$ in time $[t, t+P]$ that are not yet received. These urgent requests are made at high priority. The quality level of the urgent tiled segments is determined by the throughput estimation b_t at time t. b_t is calculated as the weighted moving average of the download size over the time period P, where the weight of the latest measurement is 0.9, if not otherwise specified. Given b_t, we set the urgent tiled segments at the highest possible quality level, while ensuring these tiled segments can be downloaded within the urgent window P. If the total segment size of the

[3]Probably because there are quite a few sudden head movements.

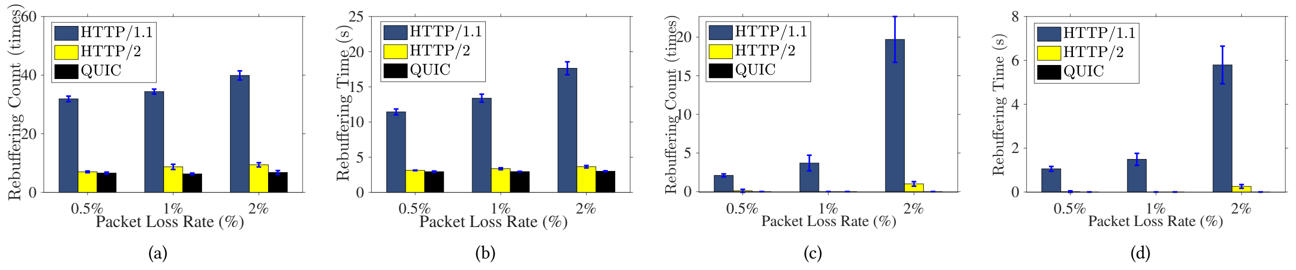

Figure 3: Comparison among protocol stacks under 5 Mbps network bandwidth: (a) rebuffering counts and (b) rebuffering time; under 8 Mbps network bandwidth: (c) rebuffering counts and (d) rebuffering time.

lowest quality level still exceeds $b_t P$, we skip tiled segments from the viewport edge until the total segment size fits the throughput.

The regular tile thread is built upon existing ABR algorithms [13] and we employ a popular buffer-based algorithm [12] in our discussions. Because we adopt a *strict priority scheduler* in QUIC, the urgent tiled segments can always be sent before the regular ones. Let B_o be the buffer occupancy (in time) and D be the total download time of urgent tiled segments between now and the preceding triggered time of the regular tile thread. Using D to approximate the time occupied by urgent requests in the upcoming round of regular requests, we consider $B_o - D$ as the *effective* buffer occupancy and pass it to the adopted ABR algorithm. The regular tiled segments are requested at low priority. Last, we note that the above principles can be augmented and applied to throughput-based and hybrid ABR algorithms [13].

5 EVALUATIONS

In this section, we evaluate our solution through a real testbed.

5.1 Implementations

We have implemented our solution based on: (i) *QUIC server/client* in Chromium [7], which includes a QUIC stack written in C++, (ii) *libcurl*, which is a client-side URL library written in C [5], and (iii) *AStream* [24], which is a Python-based DASH client with rate adaptation algorithms[4]. We build a DASH client with QUIC support, where a single QUIC connection carries all the prioritized streams for individual tiled segment requests. Several unique changes are needed, e.g., we extend the MPD file to include individual tile segment sizes at various quality levels, so that the ABR controller can make better decisions on rate adaptations. Our DASH client supports several streaming protocol stacks: QUIC over UDP, HTTP/1.1 over TCP, and HTTP/2 over TCP. The multiplexed mode is enabled in QUIC and HTTP/2. Notice that we use Caddy [15] as the DASH server for HTTP/1.1 and HTTP/2. We couldn't use Caddy for QUIC because its experimental implementation doesn't support stream priority in QUIC at the time of writing.

5.2 Setup

We set up the DASH server and client on two Intel i7 workstations with 16 GB RAM running Linux. The two workstations are connected by a GigE network. To emulate different network conditions, we employ `tc` [2] in Linux. We use `tcpdump` [18] to collect network traces for analysis. For fair comparisons among different protocol stacks, we employ a public dataset [16] to drive the DASH client. Among the 50 users watching 10 4K videos in the dataset, we select traces from 10 random users for evaluations. Each video (and experiment) runs for 60 s, and we report sample results from *Shark Shipwreck*, if not otherwise mentioned.

Several system parameters are varied in our experiments. We let the end-to-end bandwidth $C \in \{\mathbf{10}, 8, 5\}$ Mbps, the one-way network delay $d = \{\mathbf{10}, 50, 100\}$ ms, and the packet loss rate $L \in \{0.5, \mathbf{1}, 2\}\%$, respectively. We use bold font to indicate the default values. In addition, we let the initial buffer time be 2 s and urgent window size $P = 0.5$ s, buffer high/low watermarks $B_s = 3$ and $B_l = 1$ s by default. The viewport radius r_v is set to 55° (more details will be given when discussing Fig. 5). Last, the videos are encoded at three quality levels at {15, 10, 5} Mbps with 10×10 tiles.

We evaluate the results using the following metrics:

- *Bandwidth utilization.* The ratio between the used and total bandwidth.
- *Missing ratio.* The fraction of viewed tiles (based on the ground truth) that are not received in time.
- *Video quality.* The Viewport-PSNR (V-PSNR) [32] values based on the ground truth[5] from the traces.
- *Rebuffering counts/time.* The stall occurrences/duration due to buffer underrun.

Figures plot average results and 95% confidence intervals over 10 runs if possible.

5.3 Results

Benefits from QUIC. We first compare the DASH streaming quality over different protocol stacks: HTTP/1.1, HTTP/2, and QUIC. We focus on the performance results of rebuffering counts/time from a random user; results from other users are similar. For fair comparisons, HTTP/2 and QUIC do not enable their prioritized schedulers. We instruct our DASH client to request the tiles overlapping with the ground-truth viewports at the highest quality level. We consider different L and C values and plot the results in

[4]AStream doesn't include a video decoder, which however doesn't prevent us from using it to gather the performance results.

[5]We set the viewport radius as 50° as measured in Fan et al. [6].

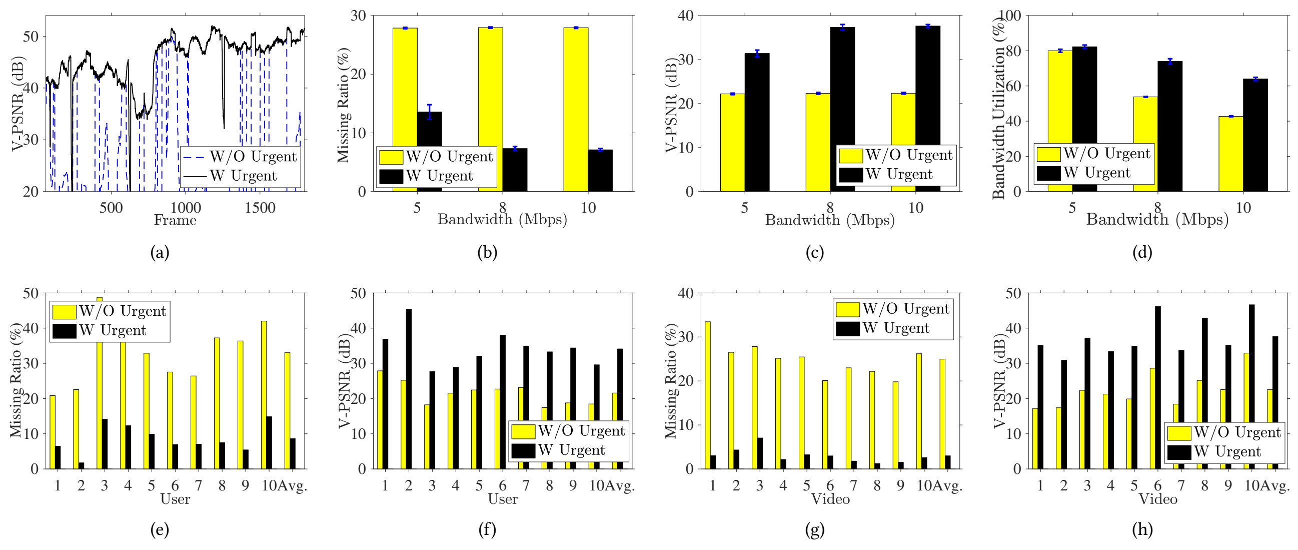

Figure 4: Effectiveness of our urgent tile streams, results from a sample user and a sample video: (a) V-PSNR over time at 10 Mbps, (b) average missing ratio, (c) average V-PSNR, and (d) average bandwidth utilization; results from different users (with all videos): (e) average missing ratio and (f) average V-PSNR; results from different videos (with all users): (g) average missing ratio and (h) average V-PSNR.

Fig. 3. This figure reveals that HTTP/1.1 gives too many rebuffering counts: as high as 39 times in each 60 s video. This can probably be attributed to the fact that HTTP/1.1 doesn't support multiplexing. Figs. 3(c) and 3(d) also confirm that HTTP/1.1 leads to high rebuffering counts/time, even when the network bandwidth is less limited. Moreover, these two figures also reveal that HTTP/2 is slightly more sensitive to packet loss rates; when the packet loss rate is 2%, HTTP/2 suffers from buffer under-run. This can be explained by the head-of-line blocking, extra retransmission packets, and longer latency caused by the underlying TCP protocol. In summary, compared to QUIC, streaming over HTTP/1.1 and HTTP/2 results in higher rebuffering counts and longer rebuffering time. Hence, we no longer consider HTTP/1.1 and HTTP/2 stacks in the rest of this paper.

Effectiveness of urgent requests. We next compare the streaming performance with and without urgent tile streams, and plot the results in Fig. 4. Fig. 4(a) presents the V-PSNR over time of a sample user on a sample video under 10 Mbps network bandwidth. This figure clearly shows the superior performance with urgent tiles in video quality. We next consider more metrics under diverse network bandwidth in Figs. 4(b)–4(d). These figures show that our urgent tile streams are useful under diverse network bandwidth: as high as 34.64% missing ratio drop and 15.29 dB V-PSNR improvement on average are observed without excessive bandwidth utilization. To understand if the above observations hold across users with diverse head movement patterns, we plot the aggregate results of individual users in Figs. 4(e) and 4(f) (bandwidth utilization is not shown due to the space limit). These two figures confirm that urgent tile streams are effective for all users. Furthermore, we also plot the aggregate results of individual videos in Figs. 4(g) and 4(h), and confirm that urgent tile streams work for all videos.

In summary, our urgent tiles effectively and constantly reduce missing ratios and increase the video quality without incurring excessive bandwidth utilization under different network bandwidth, user behavior, and video characteristics. Last, we note that no rebuffering events are observed in these experiments.

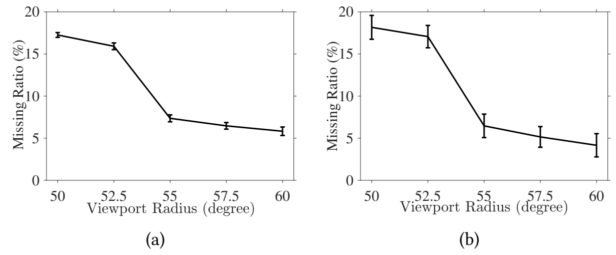

Figure 5: Missing ratios under diverse viewport radius: (a) 10 users with a sample video and (b) 10 videos with a sample user.

Compensation of imperfect fixation prediction. Among the causes of missing tiles, the imperfect fixation prediction (due to complex video content and user behavior) is the main one that may be further improved. More specifically, we next consider different viewport radius r_v values, because larger r_v results in more requested tiles, which essentially build some *cushion/buffer* zones to accommodate imperfect fixation predictions. Fig. 5 gives the missing ratios under different r_v at 10 Mbps. Fig 5(a) gives the sample results from 10 users with a video; while Fig. 5(b) gives the results from 10 videos with a user. We observe that there is a large gap between 52.5° and 55°, while the slopes beyond 55° are flatter. The average V-PSNR values (not plotted due to the space limitations) from 52.5° is at least 2.74 dB lower than that of 55°.

Approximation approach of tile selector. We compare the approximated tile selector presented in Sec. 4.2 against the ordinary one. We compute the average prediction time and the number of different tiles between them using 100 random viewports. The approximation approach terminates in 0.06 ms, while the ordinary approach takes several seconds. The average number of different tiles is 2.74, which is not serious as also evidenced in the video quality (V-PSNR) results reported above.

6 CONCLUSION

In this paper, we design, implement, and evaluate a DASH video streaming system for 360° tiled videos over QUIC. Our proposed system reduces the number of missing tiles caused by frequent viewport changes through two unique features of QUIC: *stream priority* and *multiplexing*. We conduct extensive trace-driven experiments to: (i) demonstrate the benefits of using QUIC compared to HTTP/1.1 and HTTP/2, (ii) validate the effectiveness of our urgent streams (constant superior performance under diverse network bandwidth, user behavior, and video characteristics) with up to 34.64% missing ratio drop and 15.29 dB V-PSNR increase, and (iii) determine the best viewport radius (a system parameter). The evaluation results also shed some lights on the future research directions. For example, the adaptation algorithms for the system parameters are crucial for deploying our solution in live networks. Moreover, the performance comparisons among diverse priority schedulers in QUIC and HTTP/2 should be quantitatively studied in the scope of 360° tiled video streaming. We believe this work will stimulate more research efforts on 360° tiled video (and other new content types) streaming over the future transport protocols.

REFERENCES

[1] Sevket Arisu and Ali C. Begen. 2018. Quickly Starting Media Streams Using QUIC. In *Proc. of Packet Video Workshop (PV’18)*. Amsterdam, Netherlands, 1–6.

[2] Bert Hubert. 2018. tc. (2018). Retrieved February 20, 2019 from https://linux.die.net/man/8/tc

[3] Divyashri Bhat, Rajvardhan Deshmukh, and Michael Zink. 2018. Improving QoE of ABR Streaming Sessions Through QUIC Retransmissions. In *Proc. of ACM International Conference on Multimedia (MM’18)*. Seoul, South Korea, 1616–1624.

[4] Divyashri Bhat, Amr Rizk, and Michael Zink. 2017. Not So QUIC: A Performance Study of DASH over QUIC. In *Proc. of ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV’17)*. Taipei, Taiwan, 13–18.

[5] Daniel Stenberg. 2019. libcurl. (2019). Retrieved February 23, 2019 from https://curl.haxx.se/libcurl/

[6] Ching-Ling Fan, Jean Lee, Wen-Chih Lo, Chun-Ying Huang, Kuan-Ta Chen, and Cheng-Hsin Hsu. 2017. Fixation Prediction for 360° Video Streaming in Head-Mounted Virtual Reality. In *Proc. of ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV’17)*. Taipei, Taiwan, 67–72.

[7] Google Inc. 2019. The Chromium Projects. (2019). Retrieved February 18, 2019 from https://www.chromium.org/quic/playing-with-quic

[8] Brian Hayes, Yusun Chang, and George Riley. 2017. Omnidirectional Adaptive Bitrate Media Delivery Using MPTCP/QUIC over an SDN Architecture. In *Proc. of IEEE Global Communications Conference (GLOBECOM’17)*. Singapore, Singapore, 1–6.

[9] Brian Hayes, Yusun Chang, and George Riley. 2018. Controlled Unfair Adaptive 360 VR Video Delivery over an MPTCP/QUIC Architecture. In *Proc. of IEEE International Conference on Communications (ICC’18)*. Kansas City, MO, 1–6.

[10] Brian Hayes, Yusun Chang, and George Riley. 2018. Scaling 360-degree Adaptive Bitrate Video Delivery Over an SDN Architecture. In *Proc. of International Conference on Computing, Networking and Communications (ICNC’18)*. Maui, HI, 604–608.

[11] Mohammad Hosseini and Viswanathan Swaminathan. 2016. Adaptive 360 VR Video Streaming: Divide and Conquer. In *Proc. of IEEE International Symposium on Multimedia (ISM’16)*. San Jose, CA, 107–110.

[12] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. 2014. A Buffer-based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service. In *Proc. of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM’14)*. Chicago, IL, 187–198.

[13] Jonathan Kua, Grenville Armitage, and Philip Branch. 2017. A Survey of Rate Adaptation Techniques for Dynamic Adaptive Streaming Over HTTP. *IEEE Communications Surveys Tutorials* 19, 3 (March 2017), 1842–1866.

[14] Adam Langley, Alistair Riddoch, Alyssa Wilk, Antonio Vicente, Charles Krasic, Dan Zhang, Fan Yang Fedor Kouranov, Ian Swett, Janardhan Iyengar, Jeff Bailey, Jeremy Dorfman, Jim Roskind, Joanna Kulik, Patrik Westin, Raman Tenneti, Robbie Shade, Ryan Hamilton, Victor Vasiliev, Wan-Teh Chang, and Zhongyi Shi. 2017. The QUIC Transport Protocol: Design and Internet-Scale Deployment. In *Proc. of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM’17)*. Los Angeles, CA, 183–196.

[15] Light Code Labs. 2019. Caddy. (2019). Retrieved February 19, 2019 from https://caddyserver.com/

[16] Wen-Chih Lo, Ching-Ling Fan, Jean Lee, Chun-Ying Huang, Kuan-Ta Chen, and Cheng-Hsin Hsu. 2017. 360° Video Viewing Dataset in Head-Mounted Virtual Reality. In *Proc. of ACM International Conference on Multimedia Systems (MMSys’17)*. Taipei, Taiwan, 211–216.

[17] Wen-Chih Lo, Ching-Ling Fan, Shou-Cheng Yen, and Cheng-Hsin Hsu. 2017. Performance Measurements of 360° Video Streaming to Head-Mounted Displays over Live 4G Cellular Networks. In *Proc. of Asia-Pacific Network Operations and Management Symposium(APNOMS’17)*. Seoul, South Korea, 205–210.

[18] Luis MartinGarcia. 2018. tcpdump. (2018). Retrieved February 20, 2019 from https://www.tcpdump.org/

[19] Simone Mangiante, Guenter Klas, Amit Navon, Zhuang GuanHua, Ju Ran Huawei, and Marco Dias Silva. 2017. VR is on the Edge: How to Deliver 360° Videos in Mobile Networks. In *Proc. of ACM Workshop on Virtual Reality and Augmented Reality Network (VR/AR Network’17)*. Los Angeles, CA, 30–35.

[20] Aditya Mavlankar and Bernd Girod. 2010. Video Streaming with Interactive Pan/Tilt/Zoom. In *High-Quality Visual Experience*. Springer, 431–455.

[21] Anh Nguyen, Zhisheng Yan, and Klara Nahrstedt. 2018. Your Attention is Unique: Detecting 360-Degree Video Saliency in Head-Mounted Display for Head Movement Prediction. In *Proc. of ACM International Conference on Multimedia (MM’18)*. Seoul, South Korea, 1190–1198.

[22] Minh Nguyen, Dang H. Nguyen, Cuong T. Pham, Nam Pham Ngoc, Duc V. Nguyen, and Truong Cong Thang. 2017. An Adaptive Streaming Method of 360 Videos over HTTP/2 Protocol. In *Proc. NAFOSTED Conference on Information and Computer Science (NICS’17)*. Hanoi, Vietnam, 302–307.

[23] Mirko Palmer, Thorben Kruger, Balakrishnan Chandrasekaran, and Anja Feldmann. 2018. The QUIC Fix for Optimal Video Streaming. In *Proc. of the Workshop on the Evolution, Performance, and Interoperability of QUIC (EPIQ’18)*. Heraklion, Greece, 43–49.

[24] Parikshit Juluri. 2018. AStream. (2018). Retrieved February 18, 2019 from https://github.com/pari685/AStream

[25] Qualcomm Technologies Inc. 2019. The Mobile Future of Augmented Reality. (2019). Retrieved April 13, 2019 from http://tinyurl.com/y598whjn

[26] Petrangeli Stefano, Swaminathan Viswanathan, Hosseini Mohammad, and De Turck Filip. 2017. An HTTP/2-Based Adaptive Streaming Framework for 360° Virtual Reality Videos. In *Proc. of ACM International Conference on Multimedia (MM’17)*. Mountain View, CA, 306–314.

[27] Petrangeli Stefano, Swaminathan Viswanathan, Hosseini Mohammad, and De Turck Filip. 2017. Improving Virtual Reality Streaming Using HTTP/2. In *Proc. of ACM on Multimedia Systems Conference (MMSys’17)*. Taipei, Taiwan, 225–228.

[28] Thomas Stockhammer. 2011. Dynamic adaptive streaming over HTTP –: standards and design principles. In *Proc. of the ACM Conference on Multimedia systems (MMSys’11)*. San Jose, CA, 133–144.

[29] Christian Timmerer and Alan Bertoni. 2016. *Advanced Transport Options for the Dynamic Adaptive Streaming over HTTP*. Technical Report. Alpen-Adria-Universitat Klagenfurt, Institute of Information Technology (ITEC), Austria.

[30] Mengbai Xiao, Chao Zhou, Viswanathan Swaminathan, Yao Liu, and Songqing Chen. 2018. BAS-360°: Exploring Spatial and Temporal Adaptability in 360-Degree Videos over HTTP/2. In *Proc. of IEEE Conference on Computer Communications (INFOCOM’18)*. Honolulu, HI, 953–961.

[31] Mariem Ben Yahia, Yannick Le Louedec, Gwendal Simon, and Loutfi Nuaymi. 2018. HTTP/2-Based Streaming Solutions for Tiled Omnidirectional Videos. In *Proc. of IEEE International Symposium on Multimedia (ISM’18)*. Taichung, Taiwan, 89–96.

[32] Matt Yu, Haricharan Lakshman, and Bernd Girod. 2015. A Framework to Evaluate Omnidirectional Video Coding Schemes. In *Prof. of IEEE International Symposium on Mixed and Augmented Reality (ISMAR’15)*. Fukuoka, Japan, 31–36.

[33] Alireza Zare, Alireza Aminlou, Miska M. Hannuksela, and Moncef Gabbouj. 2016. HEVC-compliant Tile-based Streaming of Panoramic Video for Virtual Reality Applications. In *Proc. of ACM International Conference on Multimedia (MM’16)*. Amsterdam, Netherlands, 601–605.