A Measurement Study of YouTube 360° Live Video Streaming

Jun Yi
Georgia State University
jyi39@student.gsu.edu

Shiqing Luo
Georgia State University
sluo10@student.gsu.edu

Zhisheng Yan
Georgia State University
zyan@gsu.edu

ABSTRACT

360° live video streaming is becoming increasingly popular. While providing viewers with enriched experience, 360° live video streaming is challenging to achieve since it requires a significantly higher bandwidth and a powerful computation infrastructure. A deeper understanding of this emerging system would benefit both viewers and system designers. Although prior works have extensively studied regular video streaming and 360° video on demand streaming, we for the first time investigate the performance of 360° *live* video streaming. We conduct a systematic measurement of YouTube's 360° live video streaming using various metrics in multiple practical settings. Our key findings suggest that viewers are advised not to live stream 4K 360° video, even when dynamic adaptive streaming over HTTP (DASH) is enabled. Instead, 1080p 360° live video can be played smoothly. However, the extremely large one-way video delay makes it only feasible for delay-tolerant broadcasting applications rather than real-time interactive applications. More importantly, we have concluded from our results that the primary design weakness of current systems lies in inefficient server processing, non-optimal rate adaptation, and conservative buffer management. Our research insight will help to build a clear understanding of today's 360° live video streaming and lay a foundation for future research on this emerging yet relatively unexplored area.

CCS CONCEPTS

• **Information systems → Multimedia streaming**.

KEYWORDS

360° live video streaming, DASH, measurement study, virtual reality

ACM Reference Format:
Jun Yi, Shiqing Luo, and Zhisheng Yan. 2019. A Measurement Study of YouTube 360° Live Video Streaming. In *29th ACM SIGMM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '19), June 21, 2019, Amherst, MA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3304112.3325613

1 INTRODUCTION

Live video streaming has played a pivotal role in shaping our lives in entertainment, surveillance and teleconference. In 2018, live video streaming is overtaking the growth of other types of online videos, with an impressive annual market increase of 113% [1]. With the emergence of 360° videos, 360° live video streaming is becoming a new way to broadcast our everyday life. Viewers are allowed to freely switch viewing directions of the panoramic content by dragging the mouse of a desktop, swiping the screen of a smartphone or moving the head around with a head-mounted display (HMD). Compared with regular live video streaming, 360° live video streaming can render a panoramic view and enrich viewer experience in various applications. Major social media websites such as Facebook and YouTube have all supported 360° live video streaming.

Despite the promising experience, achieving 360° live video streaming is challenging. Due to its panoramic nature, 360° live video streaming needs to transport high-quality data of multiple viewports and thus requires a significantly higher network bandwidth. Furthermore, the computation time for video encoding, decoding and rendering must be minimized on cameras and viewing devices to allow real-time 360° live video playback.

Facing these challenges, it is imperative to have an clear understanding of 360° live video streaming and thus to assist both viewers and system designers of this technology. For example, the viewers would like to know which video quality should be chosen when using 360° live video streaming service. For developers, an understanding of the system would help to identify the problems that exist in today's 360° live video streaming. This could be the first step to design effective algorithms to optimize the 360° live video streaming performance.

While considerable work has been done on live video streaming, they do not target 360° videos. For example, existing works studied the characteristics of regular live video streaming on different video platforms such as Skype and FaceTime [2], YouTube [3] and Facebook Live [4]. Recent efforts towards 360° video streaming generally focused on viewport-adaptive streaming [5–9], encoding/projection methods [10–12], and energy optimization [13, 14]. Although there are some measurement studies of 360° video streaming, none of them has concentrated on the measurement and analysis of 360° *live* video streaming. For example, Lo et al [15] investigated the performance of 360° video on demand (VoD) systems over 4G networks. Zhou et al [16] explored the performance of 360° VoD streaming on Oculus.

The goal of this paper is to conduct a measurement study on YouTube 360° live video streaming and to provide insight about today's commercial 360° live video streaming systems. We choose to focus on YouTube because it has over 1 billion users and is the most popular video website that can support real-time interaction with viewers during live video streaming [17]. Moreover, it is the only platform that can support 4K resolution in 360° live video streaming which has been arguably considered as the minimum resolution for panoramic viewing [18].

To achieve this goal, we develop a systematic 360° live video streaming evaluation procedure including both hardware setup

NOSSDAV '19, June 21, 2019, Amherst, MA, USA

ACM ISBN 978-1-4503-6298-6/19/06…$15.00
https://doi.org/10.1145/3304112.3325613

and measuring software development. First, we build an end-to-end YouTube-based 360° video streaming testbed to live stream panoramic content from a 360° camera to a desktop client. Dynamic Adaptive Streaming over HTTP (DASH) can be enabled in this system. Second, we design a method to automatically collect numerous metrics of the 360° live video streaming system such as video quality level, rebuffering statistics, one-way video delay, and initial delay. Third, we conduct performance analysis of the system on a variety of practical configurations including different resolutions, camera movement, and DASH support to unveil the characteristics of 360° live video streaming.

To the best of our knowledge, the proposed measurement study is the first attempt to understand commercial 360° live video streaming systems for up to 4K resolution. Our key observations can be summarized as follows.

- Today's 4K 360° live video streaming suffers from frequent rebuffering for 33% of the playback time. In contrast, the smooth playback of a 360° video at 1080p and below can be supported. However, this might imply unsatisfactory user experience in some applications since only 15% ∼ 20% of the panorama would be viewed.
- One-way video delay is extremely high in 360° live video streaming. Viewers need to wait for 13 seconds (480p) to 42 seconds (4K) to see the most recent events of the remote site. Such delay performance could be acceptable for certain broadcasting applications, but would be unsuitable for real-time interactive applications.
- While DASH can reduce the rebuffering events and one-way video delay, it still does not solve the problems of 4K 360° live video streaming. Surprisingly, viewers might not necessarily have better experience in DASH-based 4K 360° live video streaming than in non-DASH-based 1080p streaming.
- Although YouTube 360° live video streaming player has adopted a strategy to skip frames in order to reduce the one-way video delay and keep the live video up to date, the buffer management algorithm is still conservative to achieve real-time requirement.

2 RELATED WORK

Regular live video streaming. Regular live video streaming has been implemented on different platforms. Yu et al [2] conducted a measurement study of three mobile video call applications: FaceTime, Google Plus Hangout, and Skype. Through measurements over a wide range of wireless network conditions, they showed that mobile live video streaming quality is highly vulnerable to bursty packet loss and long packet delay. For the broadcast applications, Tang et al [19] described the characteristics of live video streaming on Meerkat and Periscope and revealed the relationship among different roles in the broadcast system. For example, the interaction between uploaders and viewers shapes the video content. Ding et al [3] focused on YouTube uploaders' characteristics (gender, age and geography distribution) and behavior. By analyzing 10,000 uploaders' information, they demonstrated that most uploaders prefer to upload a contiguous snippet of a video that is originally distributed outside of YouTube. Hamilton et al [20] investigated the behavior of Twitch's viewers and uploaders. It is interesting to observe that viewers tend to regard Twitch as a virtual place of social activity. Although these works have shed some light on regular live video streaming system design, the observation and principle cannot be directly used in 360° live video streaming systems.

360° video streaming. Recent efforts have been made towards the encoding/projection [10, 11], view-adaptive streaming [7, 8] and energy optimization [14] of 360° video streaming.

Meanwhile, researchers have attempted to understand 360° VoD streaming through measurement studies [13, 15, 16, 21]. Afzal et al [21] characterized 360° videos by examining thousands of YouTube videos across more than a dozen categories and reached the conclusion that 360° videos are less variable in terms of bit rate and have less motion than regular videos. Zhou et al [16] reverse-engineered the encoding solution of Oculus 360° VoD systems and identified it as offset cubic projection. Compared with the standard cubic encoding, the offset cubic projection encodes a distorted version of the spherical surface, devoting more information to the view in a chosen direction. Jiang et al [13] analyzed the energy consumption of 360° streaming on HMD under 8 test cases with different configurations and provided a detailed energy consumption breakdown of the system. They showed that the HMD streaming overhead and network transmission account for approximately half of the energy consumption. Lo et al [15] demonstrated the strength and weakness of tile-based streaming for 360° videos over 4G networks. They observed that the tile-based video streaming can clearly save the bandwidth but it may decrease the coding efficiency of tiles.

Despite the insight obtained from these studies, our understanding about 360° live video streaming is still limited. In this paper, we conduct a systematic study to facilitate viewers' interaction with this new technology as well as to lay a foundation for future system design in user experience and system performance optimization.

3 MEASUREMENT SETUP

3.1 System Architecture

There are two types of architecture for 360° live video streaming systems: direct broadcast and indirect broadcast. For the indirect broadcast, a 360° camera is used to capture panoramic images and stitch them into the equirectangular format. After being captured by the camera, the 360° video will be transmitted to a client via USB or Ethernet connection. The client can be a desktop, laptop or mobile device on which a streaming software is installed to encode the video. Then the video will be uploaded to a third-party server such as YouTube's video server. The server will transcode the video into multiple resolutions to enable DASH support and then broadcast the video to viewers. After receiving the 360° video, a player will decode the video and render the spherical images for displaying. On the other hand, for direct broadcast, the camera is connected with a router through wireless networks so that the 360° video can be directly uploaded to a third-party server without a desktop/mobile relay. These two systems have been widely applied.

In this paper, we build a direct broadcast system to transport the 360° live video. This is a more practical case in everyday life as uploaders do not need extra devices to use as the relay client when broadcasting the video. Figure 1 shows the architecture of the direct broadcast system used in our measurement study.

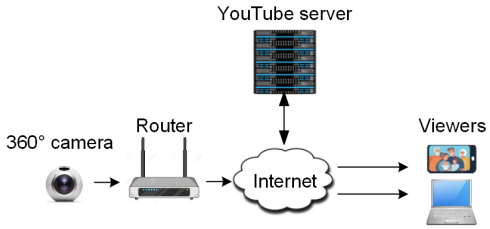


Figure 1: System architecture of the direct broadcast system.

3.2 Testbed

We build a testbed that consists of three components to study 360° live video streaming over YouTube. The 360° camera is a Ricoh Theta V, which can provide 360° live video streaming with different resolutions and bitrates. For the router, we use ASUS RT-AC3100, a dual band gigabit router. It can support 2.4 GHz and 5 GHz wireless signals simultaneously. However, we have verified that the camera only connects to the 2.4 GHz wireless service in our measurement since the router complies to 5GHz standard W58 but the Ricoh THETA V only complies to standard W52 [22]. The viewing player lies in a desktop, residing in a normal university building. For the desktop, the CPU is an Intel Xeon W-2135 with 12 cores at 3.7 Ghz, and the graphic card is a GeForce GTX 1080Ti. The high-end configuration makes it feasible to process 4K 360° live video streaming. The live stream is sent to the desktop through a university LAN connection via a Gigabit Ethernet cable. We have confirmed that the upload bandwidth from the camera to the YouTube server is 60 Mbps and the download bandwidth is 16 Mbps.

To enable the end-to-end system, we implement several system components. First, we install a plug-in [23] on the camera to establish the connection. The plug-in uses Real-Time Messaging Protocol (RTMP), which is compatible to YouTube. We input the server URL and stream key into the plug-in so that uploaders can easily live stream 360° video to YouTube. The quality of camera capture can also be configured through the plug-in. On the viewer side, we develop an 360° video player for video playback using IFrame player API [24]. The IFrame player API can embed a YouTube video player on a webpage and control the player using JavaScript. After receiving the video file chunks, our player offloads much of the processing work (decoding, audio synchronization) to dedicated video/graphic card of the computer to expedite the video processing.

3.3 Metrics and Methodology

To evaluate the performance of the YouTube 360° live video streaming service, we use several important metrics [2, 25] for live video streaming.

- *Rebuffering ratio* is calculated as the rebuffering time divided by the duration of the entire live video session. It is used to evaluate how long would the video stay in the "freeze" status.
- *Rebuffering frequency* is calculated as the total number of rebuffering events divided by the duration of the entire live video session. Although the total stalling time could be short, it is likely that the viewers may suffer from frequent rebuffering events which also degrades viewing experience.
- *One-way video delay* is defined as the time difference between when a frame is captured and when it is displayed on the screen. It measures how long it would take for a viewer to see a remote event after the event occurs. The one-way video delay includes the time for camera stitching and processing, video upload, YouTube processing, downlink transmission, video decoding, and video rendering/display.
- *Initial delay* is the time difference between when the viewer sends a request to start the live streaming to when the first frame is displayed.

Measuring these performance metrics is non-trivial. First, we have no access to the rebuffering information on the YouTube website. During the live video session, we use `getPlayerState()` function to monitor the live video streaming session. It outputs a set of values, indicating the playback status– not started, ended, paused, playing, and buffering. We also output information such as video quality and play time using `getPlaybackQuality()` and `getDuration()` functions. Based on the system logs, we can compute the rebuffering metrics.

Furthermore, it is challenging to collect the delay information since there is no API support for inserting a timestamp into each frame on the camera side. Existing tools, such as gStreamer and FFmpeg, cannot be directly applied on the camera. Even though frame timestamp is available, it is unlikely to extract the application-layer information as the video packets are encrypted by HTTPS. Therefore, we follow the methodology used in [2] to measure the one-way video delay. The key idea is to let the camera capture a stopwatch such that the camera-side timestamps become available. A second stopwatch is run in the viewing device. By measuring the difference between the two clocks when the same frame is shown, we are able to derive the one-way video delay. The configuration details can be found in [2]. For the initial delay, we can obtain it directly from the second stopwatch on the viewing client. To automatically collect the delay data, we write a script using Python 3.5 to extract the timestamps for both stopwatches.

We carry out the measurement inside a typical office building. The shooting scene generally contains professionals and office supplies such as desks and printers. As we focus on the system performance from frame capture to frame display rather than the user interaction and viewport switch, the viewer viewport does not change during the live session. The camera can capture raw video at 480p, 720p, 1080p and 4K and the viewer will pick one version upon live streaming request. The video frame rate is fixed at 30 frames per second. We perform measurement studies on various practical settings to simulate real-world application scenarios. For each video session, we measure the performance for 3 minutes and we repeat this for 20 times. The measurement scenarios are divided into four categories.

- *Default*: the camera is fixed on a table. The video with the viewer-requested resolution will be streamed.
- *Moving*: unlike Default, the uploader holds the camera and walks around the building during the measurement. Other settings remain the same.
- *DASH*: The difference between DASH and Default is that the viewer-requested video will be transcoded into more versions at the YouTube server in DASH scenario for downlink

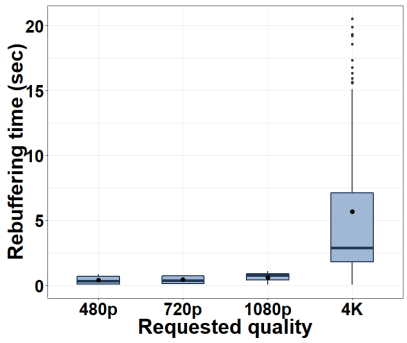


Figure 2: Distribution of rebuffering time across all sessions.

streaming (144p, 240p, 360p, 480p, 720p, 1080p, 1440p and 4K). Note that the downlink-streamed DASH videos would not have higher resolution than the viewer-requested version.

- *Moving DASH*: on top of Moving, the viewer also adopts DASH for downlink 360° live video playback.

4 MEASUREMENT RESULTS

4.1 Rebuffering Events

In this set of measurements, we measure the rebuffering events of 360° live video streaming. Figure 2 illustrates the distribution of rebuffering time of all rebuffering events across all video sessions in 4 measurement scenarios. We can observe that if the viewer-requested resolution is 1080p and below, the rebuffering time is only around 0.5s to 0.8s. We also identify in our data that the average frequency of rebuffering events is 0.6 times/min for those sessions with a viewer-requested resolution at 1080p and below. This indicates a negligible rebuffering for low-resolution live video streaming. However, if the viewer-requested video is 4K, the viewer tends to suffer from an average of 6s stall with a longest stall up to 20s. The major reason of this phenomenon is that 4K live streaming is still challenging in today's downlink networks. The rendering and processing of 360° videos further add computation burden to the end-to-end pipeline, thereby causing delayed frame delivery and frequent rebuffering.

We also individually examine the rebuffering ratio and rebuffering frequency when a 4K video is requested by the viewer in Figure 3. We observe that the rebuffering ratio is 23% and the rebuffering frequency is more than 4 times/min even under Default. In Moving, the viewer will suffer worse experience since the live video session stalls for approximately 40% of the time and the rebuffering occurs more than 6 times per minute. In fact, we have verified that the moving camera introduces unstable upload bandwidth. Furthermore, the dynamic content captured by the camera will enlarge the volume of the video, making the encoding/decoding/rendering more challenging. In addition, it is important to observe that the rebuffering ratio decreases when adaptive dynamic downlink streaming is enabled in Moving DASH and DASH. However, since the camera-captured 4K video will be transcoded into lower resolutions in DASH-enabled cases, the actual viewed video quality is far from satisfactory. We will investigate the impact of DASH in Section 4.3.

Based on our results on rebuffering events, we can conclude that viewers would experience significant rebuffering when requesting 4K 360° live video on YouTube. To ensure the smooth non-stalling

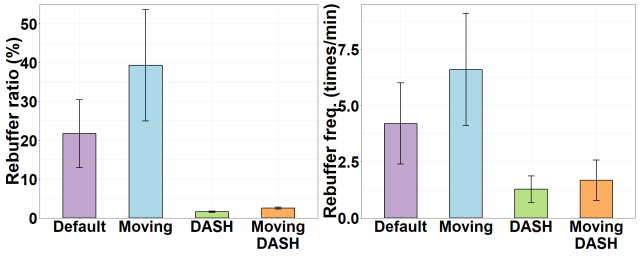


Figure 3: Rebuffering ratio and frequency of 360° live video streaming in 4K resolution.

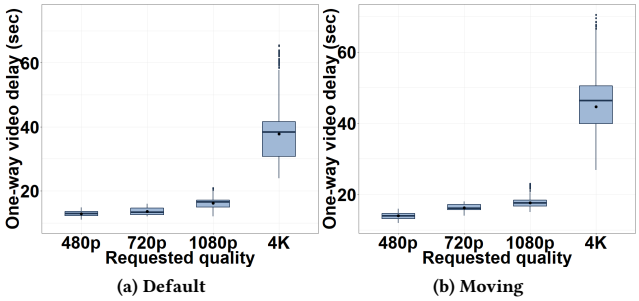


Figure 4: One-way video delay in different scenarios.

playback, viewers are suggested to request a resolution at 1080p or below. However, it is critical to note that viewer experience is likely to be unsatisfactory. This is because the resolution of 1080p or below will become even lower during viewing since only 15% ∼ 20% of the panorama would be actually viewed by the viewer.

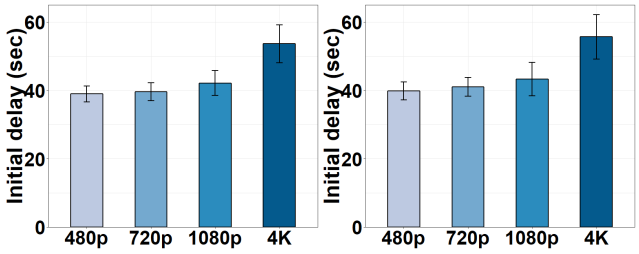
4.2 Delay Performance

In this section, we evaluate the delay of 360° live video streaming.

4.2.1 One-way Video Delay. Figure 4 shows the boxplot of one-way video delay under the scenarios of Default and Moving. We observe that the one-way video delay for 4K 360° videos is significantly higher than lower resolution. The delay range of 4K 360° live video streaming (25s to 60s) is also clearly larger. Similar to what we explained in Section 4.1, this is because each frame in 4K video contains over three times as many pixels as in lower-resolution videos. The excessive size of the video consumes more time for video transport and processing.

Another interesting phenomenon is that movement has larger impact on 4K video than lower-resolution video. While the one-way video delay of 4K video increases more than 6s in Moving, lower-resolution videos show similar one-way video delay under Default and Moving. The reason is that the network condition is significantly better than the required bandwidth for lower-resolution video. Therefore, the network speed spikes introduced by the camera movement have little impact on the one-way video delay.

4.2.2 Initial Delay. Figure 5 shows the initial delay of 360° live video streaming in different scenarios. Even for the lowest resolution, viewers need to initially wait for approximately 39s before watching the 360° live video. The start-up waiting time for 4K videos can reach 55s. We observe from our results that the large initial delay is attributed to the heavy pre-processing completed in YouTube



(a) Default (b) Moving

Figure 5: Initial delay in different scenarios.

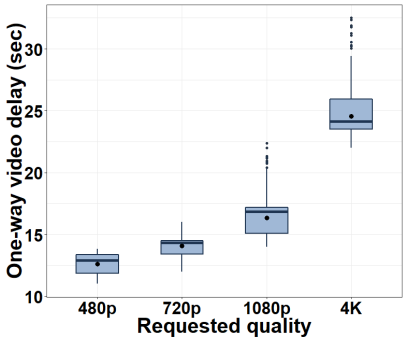


Figure 6: One-way video delay in DASH scenario.

server. In order to allow the option of DASH streaming, the YouTube server has to transcode the viewer-requested 360° video into multiple resolutions. The initial transcoding takes far more time than the subsequent frame by frame transcoding since the YouTube server initially needs to set up the encoding configuration, generate the Media Presentation Description (MPD) file, and possibly convert the projection/format of the uploaded 360° video. Therefore, we see a larger initial delay than the one-way delay in Figure 4.

From the above mentioned results, we can arrive at the conclusion that delay is probably the biggest issue for 360° live video streaming, even for low-resolution videos. Although the large initial delay and one-way video delay may be tolerant in some broadcasting applications, they are far from the requirement of real-time interactive playback. The large one-way video delay will make the viewing experience lag far behind the actual remote events. We note that server processing has played a primary role in the undesirable delay performance. New video/MPD preparation schemes at the server may be needed to expedite the 360° live video streaming.

4.3 Impact of DASH

We have shown in Figure 3 that using DASH in downlink streaming can reduce the rebuffering events. In this section, we conduct more experiments to investigate the impact of DASH on 360° live video streaming, especially for 4K resolution.

Figure 6 shows the one-way video delay of 360° live video streaming under different viewer-requested resolutions in DASH scenario. We observe that the one-way video delay is around 24.5s, which is smaller than the delay in Default and Moving scenarios. This is because DASH can dynamically change the streamed video quality to match the network condition. To further understand the viewing

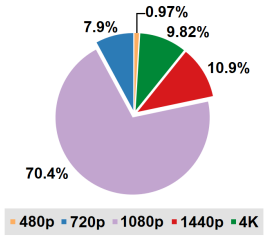


Figure 7: Resolution of DASH & Moving DASH sessions.

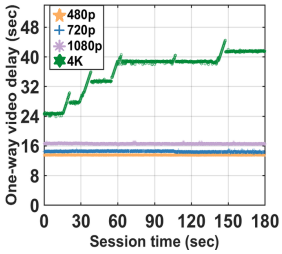


Figure 8: Per-session one way video delay in Default.

experience in DASH, we analyze the video quality actually received and viewed during the measurement sessions. Figure 7 plots the percentage of actually viewed video resolution under the viewer-requested resolution of 4K across both moving and non-moving scenarios. We observe that although the requested resolution is 4K, viewers will only watch a 4K video for 10% of the video session. Instead, viewers spend 70% of the time viewing a 1080p video. Furthermore, the viewers will to need switch among a total of five resolutions during the 3-min session. This could lead to mediocre user experience with an one-way video delay of 25s, video resolution of around 1080p, and quality variation among 5 resolutions. In fact, viewers are even suggested to directly request the non-DASH 1080p video which can achieve a one-way video delay of 18s and a constant video quality at 1080p.

The reason of this nonpositive effects of DASH on 4K 360° live video is that a conservative quality increment algorithm is adopted in the player. To further optimize the 360° live video streaming system, it is imperative for system designers to develop new rate adaptation algorithms suitable for 4K 360° live videos.

4.4 Session Trace Analysis

We have so far focused on the statistical results of the system performance. In this section, we examine a 4K 360° live video session to identify more technical issues of existing streaming solutions.

Specifically, we show the trace of one-way video delay for different viewer-requested resolutions under Default scenario in Figure 8. We see that the one-way video delay of lower-resolution videos is stable. This is expected since lower-resolution videos can be easily processed and transported in a typical computing and network environment. The large delay stems from the initial setup as we discussed in Figure 5.

For the 4K resolution, the one-way video delay continuously climbs. This is mostly due to the computation and networking constraints in handling 4K 360° live video. It is also interesting to notice that the one-way video delay of 4K video always suddenly drops after a spike and this pattern repeats across the session. After studying the trace of rebuffering status, we learn that the delay spike occurs when the player is rebuffering. Once the buffer is filled with enough video data, we observe that the player will drop a number of frames and then resume the playback, thereby leading to the sudden drop of one-way video delay. It is also worthwhile to point out that, no matter how long the rebuffering is, the one-way video delay always drops down approximately 2.5s when the live video recovers from the stall. This is because the player always

skips roughly 2.5 seconds of frames when the buffer is ready for playback.

In sum, while the existing player for 360° live video streaming does implement a scheme to skip frames in order to keep the live video up to date, the one-way video delay is too large to be acceptable. The inconsistency between the remote site and viewer will disable all interactive applications. Therefore, new buffer management algorithms are needed to address the trade-off between interactivity and scene fidelity.

5 DISCUSSION AND FUTURE WORK

Other network environment. In this paper, we only focus on measuring the performance of 360° live video streaming in Wi-Fi upload and wired download environment. Indeed, the network condition could be more diverse in practice. For example, the video may be uploaded by LTE network through a smartphone or the viewer can use a mobile HMD to download the video via wireless networks. However, we point out that the system performance of other network conditions are unlikely to be better than our setup. Hence, we believe that our observation can generally be extended to other network environment.

Other 360° live video streaming platforms. We achieve a deeper understanding of 360° live video streaming on YouTube using a Ricoh 360° camera. However, there are other platforms (Facebook and Twitch) and cameras (Samsung and Nokia) supporting 360° live video streaming. The parallel computing and media processing capability of different online platforms can be different and may impact the 360° live video streaming performance. Similarly, the stitching quality and overhead of cameras may also determines the viewer experience. In this paper, we take the first step to study commercial 360° live video streaming. A comprehensive full-scale study of 360° live video streaming pipeline is needed as future work to provide further insight on optimizing the system performance. More specific metrics will be used to further understand the system. Especially for 4K 360° live video streaming, we will try to pinpoint the component for the relatively inferior performance.

6 CONCLUSION

In this paper, we have built an end-to-end measurement testbed to investigate the performance of 360° live video streaming on YouTube. We study the system performance and user experience under various requested resolutions (up to 4K). Our observation provides important insight for both viewers and developers. Due to frequent rebuffering and annoying delay performance, viewers are suggested not to live stream 4K 360° videos, even when DASH is enabled. Although delay-tolerant applications can be achieved by broadcasting 1080p video, the excessive one-way video delay will fail any interactive application. For system designers, it is important to devise new algorithms to improve the performance of video server processing/transcoding, player rate adaptation, and player buffer management. The enhanced designs will need to maximize the video quality while keeping the streamed content up to date. We believe the results of this research can enable a suite of future works on live 360° videos, an emerging yet relatively unexplored topic in multimedia systems community.

REFERENCES

[1] Stats you need to know about live-streaming video in 2018, 2018. https://www.talkpoint.com/stats-you-need-to-know-about-live-streaming-video-in-2018/.

[2] C. Yu, Y. Xu, B. Liu, and Y. Liu. "can you see me now?" a measurement study of mobile video calls. In *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, pages 1456–1464, 2014.

[3] Y. Ding, Y. Du, Y. Hu, Z. Liu, L. Wang, K. Ross, and A. Ghose. Broadcast yourself: understanding youtube uploaders. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference (IMC '11)*, pages 361–371, 2011.

[4] O. L. Haimson and J. C. Tang. What makes live events engaging on facebook live, periscope, and snapchat. In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI '17)*, pages 48–60, 2017.

[5] C.-L. Fan, J. Lee, W.-C. Lo, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu. Fixation prediction for 360° video streaming in head-mounted virtual reality. In *Proceedings of the Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '17)*, pages 67–72, 2017.

[6] X. Corbillon, G. Simon, A. Devlic, and J. Chakaresk. Viewport-adaptive navigable 360-degree video delivery. In *IEEE International Conference on Communications (ICC)*, pages 1–7, 2017.

[7] A. Nguyen, Z. Yan, and K. Nahrstedt. Your attention is unique: Detecting 360-degree video saliency in head-mounted display for head movement prediction. In *2018 ACM Multimedia Conference on Multimedia Conference (MM '18)*, pages 1190–1198, 2018.

[8] F. Qian, B. Han, Q. Xiao, and V. Gopalakrishnan. Flare: Practical viewport-adaptive 360-degree video streaming for mobile devices. In *Proceedings of the Annual International Conference on Mobile Computing and Networking (MobiCom '18)*, pages 99–114, 2018.

[9] F. Qian, L. Ji, B. Han, and V. Gopalakrishnan. Optimizing 360 video delivery over cellular networks. In *Proceedings of the Workshop on All Things Cellular Operations, Applications and Challenges (ATC '16)*, pages 1–6, 2016.

[10] M. Xiao, C. Zhou, Y. Liu, and S. Chen. Optile: Toward optimal tiling in 360-degree video streaming. In *Proceedings of the ACM International Conference on Multimedia (MM '18)*, pages 708–716, 2017.

[11] L. Xie, Z. Xu, Y. Ban, X. Zhang, and Z. Guo. 360probdash: Improving qoe of 360 video streaming using tile-based http adaptive streaming. In *Proceedings of the ACM on Multimedia Conference (MM '17)*, pages 315–323, 2017.

[12] A. Zare, A. Aminlou, M. M. Hannuksela, and M. Gabbouj. Hevc-compliant tile-based streaming of panoramic video for virtual reality applications. In *Proceedings of the ACM International Conference on Multimedia (MM '16)*, pages 601–605, 2016.

[13] N. Jiang, V. Swaminathan, and S. Wei. Power evaluation of 360 vr video streaming on head mounted display devices. In *Proceedings of the Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '17)*, pages 55–60, 2017.

[14] Z. Yan, C. Song, F. Lin, and W. Xu. Exploring eye adaptation in head-mounted display for energy efficient smartphone virtual reality. In *Proceedings of the International Workshop on Mobile Computing Systems & Applications (HotMobile '18)*, pages 13–18, 2018.

[15] W.-C. Lo, C.-L. Fan, S.-C. Yen, and C.-H. Hsu. Performance measurements of 360° video streaming to head-mounted displays over live 4g cellular networks. In *2017 Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pages 205–210, 2017.

[16] C. Zhou, Z. Li, and Y. Liu. A measurement study of oculus 360 degree video streaming. In *Proceedings of the ACM on Multimedia Systems Conference (MMSys '17)*. ACM, 2017.

[17] Youtube help: Introduction to live streaming, 2019. https://support.google.com/youtube/answer/2474026?hl=en.

[18] M. Hosseini. View-aware tile-based adaptations in 360 virtual reality video streaming. In *IEEE Virtual Reality (VR)*, pages 423–424, 2017.

[19] John C. Tang, Gina Venolia, and Kori M. Inkpen. Meerkat and periscope: I stream, you stream, apps stream for live streams. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*, pages 4770–4780, 2016.

[20] W. A. Hamilton, O. Garretson, and A. Kerne. Streaming on twitch: Fostering participatory communities of play within live mixed media. In *in Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems (CHI '14)*, pages 1315–1324, 2014.

[21] S. Afzal, J. Chen, and K. K. Ramakrishnan. Characterization of 360-degree videos. In *Proceedings of the Workshop on Virtual Reality and Augmented Reality Network (VR/AR Network '17)*, pages 1–6, 2017.

[22] Croig Oda. Theta v client mode configuration guide, 2018. https://community.theta360.guide/t/theta-v-client-mode-configuration-guide/2565.

[23] Plug-in store, 2018. https://pluginstore.theta360.com/.

[24] Iframe api, 2018. https://developers.google.com/youtube.

[25] F. Dobrian, V. Sekar, I. Stoica, and h. Zhang. Understanding the impact of video quality on user engagement. In *Proceedings of the ACM SIGCOMM 2011 Conference (SIGCOMM '11)*, pages 362–373, 2011.