LIME: Understanding Commercial 360° Live Video Streaming Services

Xing Liu[1] Bo Han[2] Feng Qian[1] Matteo Varvello[3*]
[1]University of Minnesota – Twin Cities [2]AT&T Labs – Research [3]Brave Software

ABSTRACT

Personalized live video streaming is an increasingly popular technology that allows a *broadcaster* to share videos in real time with worldwide viewers. Compared to video-on-demand (VOD) streaming, experimenting with personalized live video streaming is harder due to its intrinsic live nature, the need for worldwide viewers, and a more complex data collection pipeline. In this paper, we make several contributions to both experimenting with and understanding today's commercial live video streaming services. First, we develop LIME (LIve video MEasurement platform), a generic and holistic system allowing researchers to conduct crowd-sourced measurements on both commercial and experimental live streaming platforms. Second, we use LIME to perform, to the best of our knowledge, a first study of personalized 360° live video streaming on two commercial platforms, YouTube and Facebook. During a 7-day study, we have collected a dataset from 548 paid Amazon Mechanical Turk viewers from 35 countries who have watched more than 4,000 minutes of 360° live videos. Using this unique dataset, we characterize 360° live video streaming performance in the wild. Third, we conduct controlled experiments through LIME to shed light on how to make 360° live streaming (more) adaptive in the presence of challenging network conditions.

CCS CONCEPTS

• **Information systems → Multimedia streaming**; • **Computing methodologies → Virtual reality**.

KEYWORDS

Live Video Streaming; 360° Video; Crowd-sourced Measurement.

ACM Reference Format:
Xing Liu, Bo Han, Feng Qian, and Matteo Varvello. 2019. LIME: Understanding Commercial 360° Live Video Streaming Services. In *10th ACM Multimedia Systems Conference (MMSys '19), Jun. 18–Jun. 21, 2019, Amherst, MA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3304109.3306220

1 INTRODUCTION

The past few years have witnessed exciting technological advances: high-speed LTE access has become a norm, mobile devices are unprecedentedly powerful, VR/AR has eventually stepped out of the lab, *etc.* Because of these advances, innovative applications such as live streaming of 360° personalized videos are now a reality.

Assume Jessie is attending a wedding ceremony of her best friend. She brings an omnidirectional camera attached to her smartphone and performs *live panoramic streaming* at the ceremony; the video is streamed to Jessie's Facebook timeline where her friends can watch the 360° view of the event in real time. Some viewers can also enjoy a fully immersive experience by using commodity VR headsets. Compared to regular live streaming, 360° live video streaming is more challenging due to its panoramic nature. It requires much higher bandwidth to provide the same perceived quality as regular video streaming. In addition, 360° video streaming has more stringent Quality of Experience (QoE) requirements to prevent VR motion sickness, and it incurs higher workload across all entities: *broadcasters*, *streaming infrastructure*, and *viewers*.

This paper provides insights from today's commercial live video streaming services. The overall goals are two-fold. First, due to a lack of measurement tools, we develop a measurement system called LIME (LIve video MEasurement platform), which allows researchers to conduct crowd-sourced measurements on commercial or experimental live streaming platforms. Second, we present, to the best of our knowledge, a first study of 360° personalized live video streaming on commercial platforms. We select YouTube and Facebook as the target platforms given their popularity.

We begin with developing LIME, a measurement system for live video streaming (§3). LIME automates the operations of one or multiple broadcasters so that they can stream pre-recorded videos – which enables repeatable experiments – via both commercial (Facebook, YouTube, Periscope, *etc.*) and experimental live streaming services. LIME allows recruiting crowd-sourced viewers via Amazon Mechanical Turk (AMT) [1], today's most popular crowd-sourcing platform. The crowd-sourced viewers are instructed to install a Chrome extension and watch live video feeds through the platform under test. The extension collects key performance statistics while the viewers watch the live videos streamed from broadcasters under LIME's control. Note that LIME itself is a generic measurement system that can work with both 360° and non-360° live video broadcasting on a wide range of streaming platforms.

We leverage LIME to collect crowd-sourced measurement data from commercial live 360° video streaming services (§4). Specifically, we deploy LIME and use AMT to recruit 548 viewers from 35 countries. Our crowd-sourced viewers watched more than 4,000 minutes of 360° live videos on YouTube and Facebook, providing a unique

* Work done while at AT&T Labs – Research.

MMSys'19, June 18–June 21, 2019, Amherst, MA, USA

ACM ISBN 978-1-4503-6297-9/19/06…$15.00
https://doi.org/10.1145/3304109.3306220

viewer-side dataset. We then conduct a comprehensive measurement study using this dataset. We make several key observations regarding live 360° video streaming on today's commercial platforms (§5). (1) Overall, the quality is not high, with 34% (35%) of YouTube (Facebook) sessions having an average *panoramic* quality no higher than 720p. Because only around 15% of a panoramic scene is typically viewed, this translates to a much lower *user-perceived* quality comprised between 240p and 360p. (2) Both streaming platforms are afflicted by long stalls, with 47% (52%) of YouTube (Facebook) sessions stalling for at least 5 seconds per minute. Surprisingly, we find such stalls are usually not attributed to the network; instead they are likely caused by the client-side computation/rendering overhead. (3) We develop a novel methodology for measuring the broadcaster-to-viewer (B2V) latency, which we find to be non-trivial for both YouTube (median 37.1 seconds) and Facebook (median 18.7 seconds); low-throughput sessions may have B2V latency of up to 5 minutes.

To complement the crowd-sourced study, we further leverage LIME to conduct controlled experiments over YouTube and Facebook to shed light on bringing *adaptiveness* to both the viewers and the broadcaster (§6). On the viewer side, we consider performing *viewport-adaptive* streaming where the server only streams content in the (predicted) viewport of a viewer. We find that performing viewport-adaptive streaming effectively reduces the video encoding bitrate by 75% to 80%. However, surprisingly, doing so does not necessarily improve the video quality on Facebook and YouTube due to several practical issues such as cautious rate adaptation design and a limited number of quality levels. On the broadcaster side, we use live videos and real-world cellular uplink traces to demonstrate the potential of *adaptive upload*, a unique optimization for personalized live 360° video streaming where the broadcaster adaptively shrinks the view being uploaded from a full panorama to smaller regions when the uplink bandwidth is insufficient. This approach has potentials of significantly reducing the stall duration (up to 99.9% reduction in our experimented scenario) and the B2V latency (up to 38% reduction).

To summarize, we make the following contributions:

- **The LIME System (§3).** We develop LIME, a generic, holistic, and crowd-sourced measurement system for live videos. LIME can be used in conjunction with the majority of today's commercial live video streaming platforms.
- **Crowd-sourced Measurement (§4, §5).** Leveraging LIME, we collect data from 548 users in 35 countries, and use this dataset to examine 360° live video streaming performance in the wild.
- **Controlled Experiments (§6).** We quantify the impact of viewport adaptiveness on 360° live video streaming. We identify inefficiencies of commercial platforms that diminish the benefits of viewport adaptiveness.

Overall, our measurement study via LIME reveals the performance of today's popular platforms for personalized live 360° video streaming. Our findings also suggest various improvements on their performance and resource utilization.

2 RELATED WORK

To the best of our knowledge, few previous work, if any, have measured or analyzed 360° live video streaming. However, there is a large corpus of work in three related areas: *personalized live streaming*, *360° video streaming*, and *crowd-sourced VR experiments*.

Personalized Living Streaming broadcasts users themselves and their surroundings using their (mobile) devices; viewers all over the world can watch the live feed in real time. In 2016, two papers [38, 43] simultaneously studied Periscope and Meerkat, back then the most popular platforms for personalized streaming on Android and iOS mobile devices. Although different in their methodologies, these two papers have a similar goal: shed some light on the architecture (*e.g.,* protocols and settings), the scale (*e.g.,* the number of broadcasters), and the performance (*e.g.,* video quality and stalls) of both streaming platforms. Our paper shares a similar goal but in the context of 360° live video streaming offered by two different platforms, YouTube and Facebook. Accordingly, our methodology largely departs from the approaches proposed in [38, 43], as well as our observations. When possible, we compare our findings with theirs. Another recent paper [42] studied the content and human factors for Periscope and Meerkat.

Twitch is another popular platform for personalized live streaming, with its primary focus on gaming broadcasting. Twitch differs from Periscope and Meerkat since its broadcasters are mostly not mobile. Pires *et al.* [33] were the first to look into Twitch (and YouTube Live) back in 2015. Using a three-month dataset, they showed the importance of Twitch with traffic peaks at more than 1 Tbps and millions of uploaded videos. Also in 2015, Zhang *et al.* [50] preliminarily investigated Twitch's infrastructure using both crawled data and captured traffic of local broadcasters/viewers. More recently, Deng *et al.* [17] expanded the latter study by exploring Twitch's infrastructure via a network of free proxies located worldwide. They identified a geo-distributed infrastructure with fine-grained server allocations, *i.e.,* resources are dynamically allocated to live streaming events based on their (growing) popularity. There are also some earlier measurements on other live streaming platforms such as live Internet TV and P2P live streaming [21, 24, 26, 39, 40]. Our study largely departs from them since we focus on different content (360° live videos) and platforms (YouTube and Facebook).

360° Video Streaming has become a hot research topic recently. Researchers have investigated multiple aspects including projection/encoding methods [6, 10, 25, 31, 52], energy consumption [23], viewport-adaptive streaming [12, 13, 15, 16, 20, 32, 35, 36, 45–47], cross-layer interaction [41, 48], and user experience [14], *etc.* Most of the above studies focused on non-live 360° videos and none of them investigated commercial 360° video streaming platforms as we have done using crowd-sourcing.

In 2017, Afzal *et al.* [11] studied the characteristics of (non-live) 360° videos uploaded to YouTube by simply *searching* for such videos using keywords like "360". By analyzing a dataset of 4570 videos, they found that compared to regular videos, 360° videos tend to be shorter, having higher resolutions, and more static (less motion). Our work complements this effort since we focus on the streaming performance rather than the 360° video characteristics.

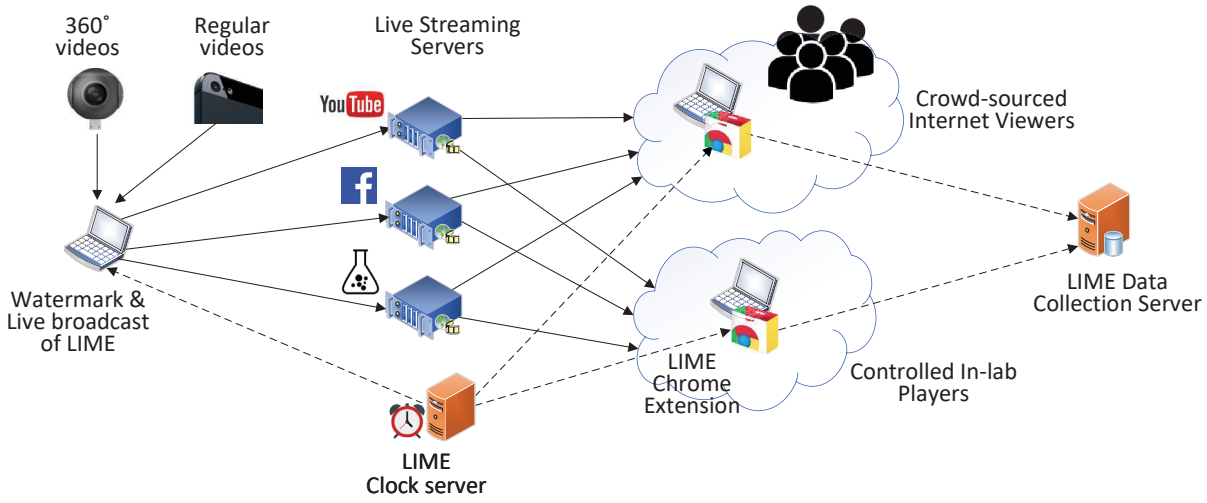

Figure 1: The system architecture of LIME.

Furthermore, we investigate live streaming and expand our analysis to Facebook as well.

360° live video streaming is a natural evolution of personalized live video streaming as fueled by the prevalence of 360° cameras. Support for 360° videos was recently added to Periscope, YouTube, and Facebook, to name a few. In 2017, a positioning workshop paper conducted a preliminary investigation of live 360° videos [27]. Our study goes beyond [27] by making several contributions: developing a holistic measurement system for live videos, conducting crowd-sourced measurements for live 360° videos, and quantifying the benefits of several key optimizations.

Crowd-sourced VR Experiments. A recent study leveraged Amazon Mechanical Turk (AMT) to recruit participants who were equipped with VR headsets to conduct behavioral experiments for VR [28]. In our study, we also recruited voluntary participants on AMT though we targeted regular desktop users and 360° videos instead of VR. Nevertheless, we have leveraged several key insights of [28] to facilitate our crowd-sourced experiments.

3 LIME DESIGN AND IMPLEMENTATION

Live video streaming is a popular technology that allows a *broadcaster* to share a video (s)he is recording in real time with *viewers* all around the world. Measuring and experimenting with live video streaming is challenging due to several reasons. First, the time constraint of live videos makes it hard to build reproducible experiments. Second, by nature the system requires live viewers. However, access to viewers is not trivial; measuring video statistics from live viewers scattered all around the world can be challenging. Third, there exist other challenges from a system's perspective, such as logging key performance statistics in a lightweight manner on unmodified browsers and accurately measuring the broadcaster-to-viewer delay.

In this section, we describe LIME (LIve video MEasurement platform), a measurement platform for live video streaming. Our (ambitious) goal is to build a generic platform that enables measuring existing live video streaming systems (*e.g.,* Facebook or YouTube) from worldwide viewers using real video feeds, as well as testing new streaming algorithms and/or infrastructures. In this paper, we use LIME to answer two research questions. First, what are the key performance characteristics of live 360° video streaming services on YouTube and Facebook (§5)? Second, what is the impact of novel streaming techniques on these commercial platforms (§6)? In addition, we can leverage LIME in many other contexts to facilitate the multimedia research and to gain insights from large-scale, production-grade live video streaming systems. Several example studies that could be enabled by LIME but are beyond the scope of this paper include: quantitatively comparing 360° and non-360° live videos, measuring users' engagement levels for different types of 360° live content, and reverse-engineering the rate adaptation algorithms of commercial live streaming platforms.

3.1 Overview

Figure 1 shows LIME's key components: the *broadcaster*, *streaming servers*, a set of *viewers* (either crowd-sourced viewers or controlled players), the data collection server, and the clock server. The broadcaster is a Linux machine instrumented with Open Broadcaster Software (OBS) Studio [7] to broadcast pre-recorded videos to the target streaming servers, either commercial or controlled by the "experimenter" *i.e.,* a researcher using LIME for her own studies. Such a "replay" approach ensures repeatable experiments. The vast majority, if not all, of popular live streaming services (*e.g.,* YouTube, Facebook, Periscope, and Twitch) currently support OBS and can be thus tested via LIME. The experimenter can use either a regular camera to shoot non-360° videos, or a panoramic camera (*e.g.,* Insta360 [4]) to capture 360° videos. The camera can be either stationary or mobile, depending on the content to capture and the experimental setting. Also, both Facebook and YouTube (and possibly other platforms) allow to perform a live broadcast publicly by setting the channel to "public", so the viewers can directly watch the feed through a simple URL without having to log into their

Item	API/Obj	Interval
Video Quality Change	`HTML Player`	Event Triggered
Rebuffering Events	`HTML Player`	Event Triggered
HTTP Req/Res Headers	`chrome.debugger`	Event Triggered
User Interactions	`HTML Window`	Event Triggered
Video Screenshots	`chrome.tabs`	Every 2 Seconds

Table 1: Data items collected by LIME.

accounts. This alleviates the privacy concern as the broadcaster has no way to know the identities of the crowd-sourced viewers.

The viewers consist of both *crowd-sourced* Internet viewers and *in-lab* players. To allow recruiting viewers in a short period of time, we integrate LIME with Amazon Mechanical Turk [1], a popular crowd-sourcing platform. The experimenter can thus select a number of target viewers (with specific locations/demographics if needed), how many videos they should watch, and their compensation. In-lab players consist of Chrome's instances equipped with LIME's Chrome extension (described next) which are automated via Selenium.[1]

LIME collects viewing statistics using a Google Chrome extension (1,000 lines of JavaScript). We choose a browser extension because it is lightweight, secure, and easy to install. Running in the background on the viewer side, the extension collects the following data summarized in Table 1: (1) the video playback quality (*e.g.,* 720p), (2) stall (*i.e.,* rebuffering) events, (3) HTTP request/response headers, (4) user interactions such as dragging the mouse to change the viewport, and (5) periodically captured (every 2 seconds) screenshots of the visible portion of the video.

LIME captures these data items from either Chrome extension APIs (`chrome.debugger` and `chrome.tabs`) or HTML DOM objects (`HTML Player` and `HTML Window`), as listed in Table 1. After the video playback finishes, the data is automatically uploaded to the LIME data collection server over HTTPS for further offline analysis. We verified that the extension incurs negligible runtime overhead. We also engineered the extension to be robust to various scenarios such as switching to a different tab while the video is playing. Note that our Chrome extension is a versatile measurement tool. It works for any video platform (both live/non-live and 360°/non-360°) realized by the HTML Video DOM [3], a popular HTML-based video solution.

Finally, Figure 1 shows that LIME further includes two servers under the experimenter's control: a *data collection server* and a *clock server*. The data collection server is the end-point where users' viewing statistics collected by the Chrome extension are uploaded to. The clock server provides synchronized clock readings to both the broadcaster and the viewers, in order to enable accurate measurement of the broadcaster-to-viewer latency. More details will be provided in §3.2.

3.2 Measuring the Broadcaster-to-Viewer (B2V) Latency

B2V latency is an important QoE metric for live streaming. We define it as the latency from when a frame leaves the broadcaster to when the frame is consumed by a viewer. A long B2V latency causes lags that are undesirable for real-time live events such as sports. One possible methodology of measuring B2V latency is as follows. The broadcaster watermarks every frame f with a timestamp $t_B(f)$ denoting the time when the frame is generated. When the same frame f is being played, the viewer obtains the current timestamp $t_V(f)$ from the clock server (Figure 1). Meanwhile, $t_B(f)$ can be extracted from the frame (we use Tesseract OCR [8] to perform text recognition on screenshots). The B2V latency for f is therefore $t_V(f) - t_B(f)$. Note that $t_B(f)$ is obtained from the same clock server as $t_V(f)$. Also note that the originally obtained $t_B(f)$ and $t_V(f)$ need to be calibrated by subtracting the one-way latency to the clock server (estimated as half of the measured RTT)[2].

The above scheme works for non-360° live videos. However, we face two practical challenges when applying it to 360° live videos. The first challenge relates to *projection* used in 360° videos. The OBS software can apply the watermark to only *unprojected* raw frames that contain the panoramic 360° views. During a playback, a viewer's browser will apply the corresponding projection algorithm (*e.g.,* Equirectangular [2] for our videos used in §4.2) to display the visible portion. After projection, the timestamp watermark may be distorted, making OCR difficult. To address this, we embed the watermark at a special spot (latitude = 0° and longitude = 0°) to minimize the distortion for equirectangular projection. Similar spots can be identified for other projection schemes.

The second challenge is that the Chrome Extension API allows our data collector to capture only the *visible* portion of a panoramic frame that may not contain the watermark. A possible solution is to embed multiple watermarks that cover all possible viewing areas, but doing so will affect viewers' viewing experiences and make OCR challenging again due to distortions incurred by projection. Instead, we introduce a *helper player*, a special client controlled by us, which always "looks" at a fixed direction (*i.e.,* latitude = 0° and longitude = 0°) whose FoV contains the watermark. During a live broadcast session, the helper player continuously extracts $t_B(f)$ from the received frames and sends $t_B(f)$ to the data collection server. Note that for each frame, its $t_B(f)$ only needs to be extracted once regardless of the number of viewers, so we need only one helper player per broadcaster. When an actual viewer receives f, it does not need to perform watermark extraction; it only needs to record its own $t_V(f)$ and send it to the data collection server, which can now compute the B2V latency for frame f watched by this *particular* viewer as $t_V(f) - t_B(f)$. Note that for both the helper player and actual viewers, their communication with the data collection server can be performed offline if we do not need to know the B2V latency in real time (§4.2). In this case, the helper player or a viewer will upload all frames' $t_B(f)$ ($t_V(f)$) values to the data collection server in a single batch. In fact, in this scenario, even the OCR analysis can be performed offline to make the system more flexible.

3.3 Limitations

We discuss several limitations for our current LIME system.

[1] https://www.seleniumhq.org/

[2] It is possible to use other off-the-shelf clock synchronization protocols to provide more accurate timestamps. Our B2V measurement technique can work with any clock synchronization protocol.

Broadcasters are an important component of commercial live video streaming. Monitoring their behaviors, *e.g.,* when, what, and how frequently they broadcast, can be useful to improve the performance of a live streaming service. LIME allows an experimenter to *setup* multiple concurrent broadcasters, but it does not allow to *monitor* existing broadcasters "in the wild". A possible way to measure the broadcasters in the wild is to utilize platform-specific APIs. For example, some existing studies [38, 43] utilize Periscope's APIs to crawl its broadcasters. However, no similar APIs exist for YouTube and Facebook.

Our current LIME system captures important data items as listed in Table 1. Despite this, several other types of data such as network traffic and lower-layer (*e.g.,* TCP) events, which can enable cross-layer analysis, are not collected by LIME at this moment. The reason is that collecting them requires heavy instrumentation and may incur privacy concerns. While this is not an issue for experimenter-controlled in-lab players, it is a concern in presence of crowd-sourced viewers.

Last but not least, LIME can be further extended in several ways such as collecting users' subjective ratings of the viewing experiences, as well as calculating other objective quality metrics such as Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM) [44], and Video Multimethod Assessment Fusion (VMAF) [51]. We will leave them as future work.

4 COLLECTING CROWD-SOURCED DATA FOR LIVE 360° STREAMING

This section describes how we leverage LIME to collect crowd-sourced measurement data from commercial live 360° video streaming services. We start with the background on 360° video streaming, and then dig into LIME's specific setup for this measurement campaign.

4.1 Background

360° videos differ from regular videos in that a viewer can freely change her viewing direction during a streaming session. As of today, popular streaming platforms supporting 360° videos, *e.g.,* YouTube, Facebook, and Periscope, take a *monolithic* approach by transmitting the entire panoramic scene. The client-side player then applies projection algorithms such as Equirectangular [2] or CubeMap [10] to display only the visible portion of each frame to a viewer. As a result, in monolithic 360° video streaming, the viewer consumes only a fraction of the actual downloaded content.

A live 360° video streaming system typically consists of three components: a *broadcaster*, the *streaming server*, and multiple *viewers*. The broadcaster uses an omnidirectional camera to shoot panoramic videos and uploads them to the streaming server, which then re-encodes them using multiple quality levels, *e.g.,* 144p to 1440p for YouTube and 720p/1080p for Facebook, in real time. The viewers fetch from the streaming server video chunks at the quality chosen by a rate adaptation algorithm running on their devices.

For this study, we target commercial 360° live streaming systems. We select two platforms, YouTube (YT) and Facebook (FB), due to their popularity and a lack of previous studies differently, for example, from Periscope [38, 43]. Prior to the study, we perform experiments with full control on both viewer(s) and broadcaster to understand how the two platforms work. More specifically, we use various tools such as Chrome developer tools, tcpdump, and mitmproxy [5] (for decrypting HTTPS traffic). The experiments were conducted in May 2018. We find that YT and FB use Adobe's RTMP (Real-Time Messaging Protocol [10]) for live video upload. They use RTMP differently though. YT employs RTMP over TCP directly on port 1935, while FB tunnels RTMP traffic inside HTTP/HTTPS. For both YT and FB, the broadcaster performs no rate adaptation, *i.e.,* a video is uploaded at the quality determined by the broadcaster regardless of network conditions. On the download path (server to viewers), they both employ *polling-based DASH-style rate adaptation*. Specifically, a viewer periodically polls – every 5 seconds for YT and every 1 second for FB – for a Media Presentation Description (MPD) file, which contains meta data such as the URLs and quality levels for chunks that have been recently generated (re-encoded) by the server. The viewer then selects an appropriate quality level for each newly generated chunk (if any) and fetches it over HTTPS. We find the chunk duration equals to the polling interval: 5s for YT and 1s for FB. Overall, the above scheme is different from Periscope live streaming that uses a combination of RTMP and HLS (HTTP Live Streaming) protocols on the download path, as reported by a previous study [43].

4.2 Data Collection Using LIME

We now detail the data collection procedure. Using a panoramic camera (Insta360 [4]) attached to a smartphone, we shoot three 10-minute long 360° videos: (1) a city street view shot by mounting the camera on a car, (2) an on-campus walk shot by hand-holding the camera, and (3) a bicycle racing game shot with a stationarily placed camera. We believe that these videos represent typical live 360° video content. In the following, we refer to them as *Street*, *Campus*, and *Racing*, respectively.

We use LIME's broadcaster to stream, overall, the three videos (*Street*, *Campus*, and *Racing*) per platform (YT and FB), for a total of 6 live feeds. To guarantee the absence of bandwidth bottleneck on the upload path, we periodically use `testmy.net` [9] to measure the upload bandwidth between LIME's broadcaster and YT/FB servers. We find that the upload path always has more than sufficient bandwidth (>50 Mbps).

We invite Internet users to participate in our IRB-approved study by installing LIME's Chrome extension and watching live videos. Viewers can watch multiple live videos, but they are restricted to one video at a time and each video at most once. During the live video streaming, a viewer can freely change the viewport by interacting with the player (*e.g.,* dragging the mouse or swiping on the touchscreen). We ask the viewer not to change the default *auto* quality option in the player so that we can study YouTube and Facebook's built-in rate adaptation mechanism. Viewers are also required to watch a video for at least 4 minutes, after which they can end their task. Only at this time, the extension uploads the collected data to our server, without impacting the live video streaming experience.

During a 7-day study in May 2018, we kept our (replayed) live video feeds alive, and used Amazon Mechanical Turk (AMT) to recruit 548 paid viewers from 35 countries, with USA and India contributing 78% of the viewers, as shown in Figure 2. If a viewer

Figure 2: Geographical distribution of paid AMT users.

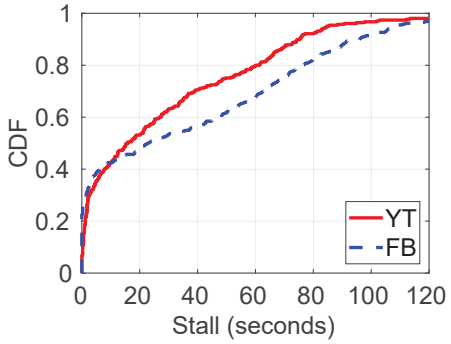

Figure 4: Distributions of per-session stall duration.

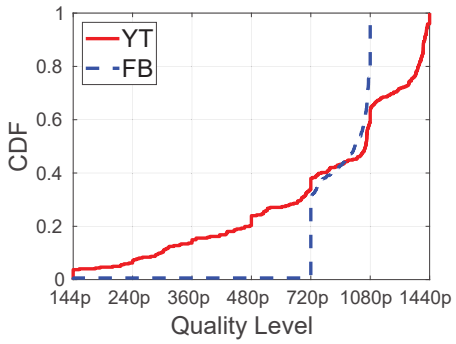

Figure 3: Average streaming quality across all sessions.

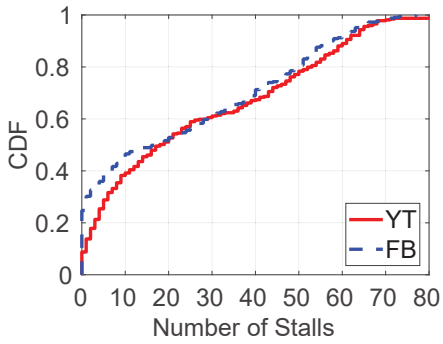

Figure 5: Distributions of per-session number of stalls.

follows the previous instructions, she/he is compensated with $0.8. Overall, we collected 22 GB data corresponding to more than 4,000 minutes of video viewing. Many paid viewers watched our feeds for more than 4 minutes. To prevent bias toward viewers with long viewing time, our analysis focuses on only the first 4 minutes per viewer. During the study, no personally identifiable information (PII) was collected.

5 UNDERSTANDING COMMERCIAL LIVE 360° VIDEO STREAMING IN THE WILD

We now characterize the data collected in §4.2 to reveal the landscape of the performance of today's popular live 360° video streaming services.

5.1 Basic Viewer-side QoE Metrics

It is known that three key factors affect the QoE of regular video streaming (both live and on-demand): video quality, stall duration, and quality changes [18, 22, 29, 49]. These metrics are also important to panoramic live video streaming so we quantitatively measure them using the AMT dataset. We find that our three videos yield very similar distributions on all metrics. Thus, we present their aggregated results henceforth.

• **Video Quality.** Figure 3 plots the Cumulative Distribution Function (CDF) of the average video quality for YT and FB across all viewing sessions. Recall that each viewing session is a 4-minute 360° live video content streamed to one paid AMT viewer. As shown in this figure, a key difference between YT and FB is that the average quality for most FB viewers is either 720p or 1080p, which are the only two quality levels provided by FB; in contrast, YT supports 7 quality levels ranging from 144p to 1440p, providing more flexibility as we will discuss shortly. Overall, the live 360° videos' quality is not high, with 34% (35%) of YT (FB) sessions having an average quality no higher than 720p. It is important to note that the above qualities refer to the qualities of panoramic frames. Since a viewer perceives only a small portion of the panoramic scene (about 15% as measured in §6.1), the actual *perceived* quality is much lower, *e.g.,* 15% of the 720p resolution is between 240p and 360p.

• **Stall.** Figure 4 plots the CDFs of the stall duration per session. About 36% (39%) of YT (FB) sessions experience less than 5 seconds stalls. However, many users experience very long stalls: 47% (52%) of YT (FB) sessions stall for at least 20 seconds (*i.e.,* 5 seconds per minute). FB is afflicted by longer stalls than YT, despite their viewing sessions characterized by similar throughput distributions. Given this, we are leaning towards attributing YT's shorter stall duration to its wider range of quality levels, which allow YT players to flexibly select (lower) streaming bitrates.

We also investigate how stalls are distributed within a session. We find that the vast majority of stall events, which are separated using an inter-stall time of at least 1 second, are fairly short: for YT (FB), the median of their durations are 0.7 (1.0) second. Note that

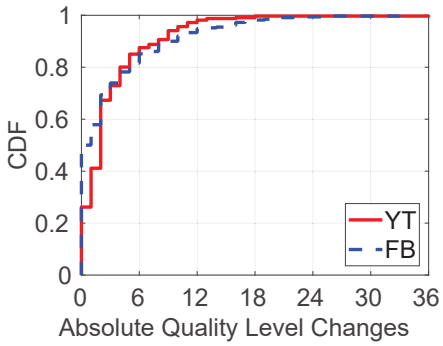

Figure 6: Distributions of per-session quality level changes.

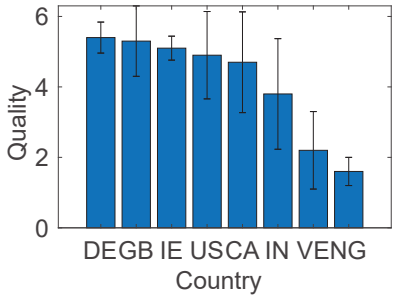

Figure 7: Distributions of average quality levels across viewers in different countries for YT (DE: Germany, GB: United Kingdom, IE: Ireland, US: United States, CA: Canada, IN: India, VE: Venezuela, NG: Nigeria).

changing the threshold of inter-stall time to 0.5s or 1.5s yields similar results. Meanwhile, the number of stall events is large. Figure 5 plots the CDFs of the number of stall events per session, with the median measured to be 18 (17) for YT (FB), respectively. Compared to fewer long stall events, having more short stall events may bring even worse QoE (*e.g.,* motion sickness in VR) [19, 30, 34]. In fact, users who experience long or frequent stalls as exhibited in the tails in Figure 4 and Figure 5 will likely abandon watching the videos. Such abandonment behaviors are not captured by our current study where users are required to watch a live video for at least 4 minutes in order to get paid. This is a limitation of our study. We plan to study 360° video viewers' abandonment behaviors in our future work.

Since the stall duration shown in Figure 4 appears to be much higher than those measured by a previous study on Periscope [38], we attempt to find out the reason. Recall from §4.1 that live video streaming consists of four phases: upload, server-side re-encoding, download, and client-side rendering/playback. For upload, we ensure high bandwidth between our broadcaster and the streaming servers. For download, surprisingly, we observe very little correlation between the per-session stall duration and its network throughput or throughput variation (Pearson correlation coefficient < 0.1). In fact, many viewers have very high network throughput but still experience high stalls (recall that YouTube can reduce its quality to as low as 144p). For example, one-third of the top 30% YT sessions in terms of the stall duration are characterized by an average throughput ranked in the top 30% of the sessions.

The above observation makes us believe that for live 360° video streaming, the performance bottleneck is caused by either the server processing (real-time video re-encoding and possibly projection transformation) or client-side computation/rendering. In particular, we are able to verify that the *client-side overhead* can cause frequent stalls: when testing on a 2013 MacBook Pro, we observe the CPU utilization of 360° live streaming can reach up to 80%, about 60% higher than non-360° live streaming. When the CPU is saturated, the live streaming can oftentimes stall.

• **Quality Changes.** Figure 6 plots the CDFs of the total number of quality level changes per session. When a quality level change occurs, it is counted as $\Delta L = |L_{\text{before}} - L_{\text{after}}|$ where L_{before} and L_{after} are the quality levels before and after the change, respectively. Possible levels are {0,1,...,6} for YT and {4,5} for FB[3]. We then sum up all ΔL to get the total quality level changes per session. For most sessions, we did not observe significant quality changes: the median of $\sum \Delta L$ is only 2 and 1 for YT and FB, respectively. Nevertheless, we do find about 6% (10%) for YT (FB) of sessions with $\sum \Delta L \geq 10$, attributed to the highly variable bandwidth as confirmed from our measured network throughput.

• **Performance vs. Geographical Location.** We observe that viewers' perceived performance (and hence the QoE) oftentimes differs statistically depending on the location. Figure 7 plots the average quality levels across viewers in major countries appearing in our dataset for YT. The per-country average quality level ranges from 1.61 (between 240p and 360p) to 5.44 (between 1080p and 1440p). As somewhat expected, viewers in developed countries experience higher qualities in general, likely due to their better network and data center infrastructures, compared to viewers in developing countries. We observe similar trends for stall duration.

5.2 Broadcaster-to-viewer (B2V) Latency

We apply the method introduced in §3.2 to measure the B2V latency for the AMT viewers, with the results shown in Figure 8. We make three observations. First, most sessions have consistent B2V latencies, around 18.7 seconds for FB and 37.1 seconds for YT (median values). Overall, the observed B2V latency is much higher compared to previous measurement on Periscope [43], which uses push-based RTMP on a subset of users to provide an ultra-low B2V latency (less than 2 seconds). Second, both platforms exhibit long tails of up to 4.8 minutes for FB and 5.1 minutes for YT. Such high latency inevitably affects viewers' experience. Although it is difficult to reverse engineer the precise algorithm, we find that throughput appears to be a factor that affects the B2V latency. For example, YT exhibits a negative correlation (Pearson correlation coefficient of -0.4) between the B2V latency and throughput. Third, we also notice that FB exhibits lower B2V latency than YT. This can be explained by several potential reasons. One is that compared to YT, FB has

[3]We use the following quality mappings throughout this paper: 0=144p, 1=240p, 2=360p, 3=480p, 4=720p, 5=1080p, 6=1440p.

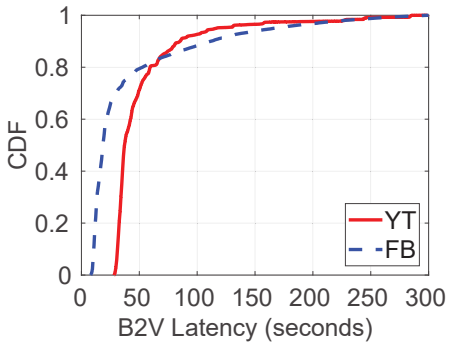

Figure 8: B2V latency of samples across all sessions. In each session, B2V latency is measured every 2 seconds.

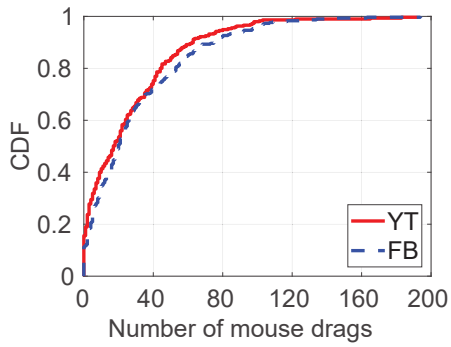

Figure 9: Distribution of mouse drags per user during a 4-minute live video.

a shorter chunk duration (1 vs 5 seconds) that leads to a lower chunking delay [43]. FB also has a lower polling delay allowing the viewer to update the chunk list more quickly. In addition, recall that a FB server only re-encodes the input video into 2 quality levels while a YT server needs to process 7 quality levels up to 1440p. Such a higher re-encoding overhead may also contribute additional latency.

5.3 User Interaction

A key feature of 360° videos is that viewers can freely change their viewing direction. Since our AMT viewers are Internet users, they typically interact with the player by mouse drags or screen swipes, which are captured by LIME's Chrome extension, as opposed to wearing a VR headset. Figure 9 shows the number of captured mouse drags or screen swipes across all sessions. We find that most of our viewers did change their viewports frequently: 46% (YT) and 51% (FB) users incur at least 20 interaction events during a 4-minute session. Note that since both YT and FB employ monolithic streaming (§4.1), the viewport changes do not affect the bandwidth usage or rate adaptation decisions.

6 CONTROLLED EXPERIMENTS

In this section, we complement our crowd-sourced study with controlled experiments to shed light on bringing *adaptiveness* to both the viewers (§6.1) and the broadcaster (§6.2). These are the key features missing from today's 360° live video streaming systems. We note that the underlying concept of viewport-adaptiveness is not new [13, 16, 20, 27, 35, 36, 46]. Nevertheless, our contribution here is to quantitatively study its benefits using commercial systems, realistic live 360° video content, and real users' viewing trajectory traces. In addition, we also identify several inefficiencies in production 360° live video streaming systems that diminish the benefits of viewport adaptiveness. We conduct all controlled experiments over LIME by replacing crowd-sourced viewers with in-lab players.

6.1 Viewport Adaptive Streaming to Viewers

Our measurements in §5 indicate that viewers of live 360° videos oftentimes experience degraded QoE. One reason for this is the monolithic streaming paradigm of YT and FB (§4.1). We thus want to understand the potential of *viewport-adaptive streaming*, a hot research topic where the server streams only content in the (predicted) viewport of a viewer [13, 35]. Developing a full-fledged viewport-adaptive 360° live video streaming system is beyond the scope of this paper. Here we are interested in the following "what-if" scenario: *how much QoE improvement can be achieved by delivering only visible portions of panoramic scenes?* This is an "optimal" scenario that leads to the lower bound of the content distribution overhead.

Lacking control on the commercial streaming servers, we resort to creating a "viewport-only" (VO) version of a 360° video, and compare its live streaming – in terms of various metrics measured in §5 – with its panoramic counterpart. This approach is *transparent* to the streaming infrastructure as the VO version can still be watched in an unmodified player of YouTube/Facebook. A challenge here is to ensure apple-to-apple comparisons between the two video versions. The detailed steps we take are as follows.

First, we conduct another IRB-approved user study involving 12 people[4] from whom we collect viewing directions when watching two live videos (*Campus* and *Street*, see §4.2). Since the Chrome extension of LIME cannot capture a user's viewing direction, we developed a separate data collector that runs on Android smartphones, allowing the viewers to more easily and naturally adjust their viewport by simply moving the phone. The viewing trajectory is captured by motion sensors. Given a 360° video and a viewer's viewport trajectory, we identify each frame's visible portion based on the projection algorithm. Next, we re-encode the original video frame-by-frame into two versions with identical encoding configuration and projection scheme (equirectangular): one without any modification (panoramic), and the other with viewer's invisible portion removed (VO). Finally, a controlled player of LIME (a Linux laptop running Chrome) "watches" these two versions as live feeds through YouTube and Facebook under the same network condition.

We now describe our findings. We begin with understanding how user diversity affects the VO videos. We find that across all users, despite their different viewing trajectories, the fractions of consumed portions are very similar: 15.0%±0.5% of all pixels in raw panoramic frames for *Campus* and 15.1%±0.3% for *Street*. This is

[4] Students and faculty from a university as well as employees in a large U.S. corporation.

Table 2: Compare panoramic (P) and viewport-only (VO) versions of the *Campus* video, in terms of average quality level (Q) and stall (S, in seconds). Each test is a 20-minute run. Q_b is the stall-free highest achievable quality level manually measured by us.

Platform/ Version	8Mbps			4Mbps			2Mbps		
	Q	Q_b	S	Q	Q_b	S	Q	Q_b	S
YT (P)	5.0	5	<1	4.0	4	<1	3.0	3	<1
YT (VO)	4.8	6	<1	4.0	5	<1	3.0	4	<1
FB (P)	4.9	5	<1	4.0	4	<1	4.0	4	29
FB (VO)	5.0	5	<1	4.9	5	<1	4.0	4	<1

because for most of the time, users keep their viewing direction leveled (*e.g.,* |latitude| <15°) and change their sight horizontally [13], *i.e.,* only adjusting the longitude which does not cause the viewport size to change in equirectangular projection. Consequently, the broadcaster-side encoded bitrates of VO videos are only 21.9%±1.7% and 25.4%±1.2% of their corresponding panoramic versions for *Campus* and *Street*, respectively. The ratios are similar for those re-encoded by YT/FB.

We next perform, under controlled settings using LIME, live broadcasting of both versions of the *Campus* video for a user whose VO video has a size closest to the median size computed among all VO videos. Note that other users/video yield similar results. Since we study the viewer side, the download bandwidth is throttled to {8, 4, 2 Mbps} to emulate different network conditions, while we ensure high upload bandwidth (>50Mbps) at the broadcaster.

We show the results in Table 2. As we reduce the download bandwidth, the rate adaptation algorithm effectively reduces the average quality level (Q) while maintaining overall low stalls (<1 second). Only for FB (P), *i.e.,* the panoramic version, we observe long stalls (29 seconds with 2 Mbps throttling). This is attributed to FB's limited quality levels (720p and 1080p) as streaming the panoramic *Campus* video at the minimum FB quality (720p) requires more (on average, 2.2Mbps) than the throttled bandwidth. These stalls are eliminated when streaming the VO version since it only consumes ~1.2Mbps of bandwidth on average.

Meanwhile, Q_b in Table 2 denotes the stall-free highest achievable quality level measured *manually* without using the automatic rate adaptation. Ideally we want Q to be equal to Q_b. For YT, we make an interesting observation that switching to the VO version provides room for quality improvement, which, however, was not actually observed: Q_b(VO) > Q_b(P), but Q(VO) ≈ Q(P). As a VO video has a much lower encoded bitrate (but the same resolution) compared to its panoramic counterpart, it appears that YT mainly uses the video's resolution for rate adaptation (*e.g.,* by assuming fixed mappings from the resolution to a video's encoded bitrate), without considering the actual encoded bitrate, leading to a lower-than-desired quality for VO videos.

Regarding FB, Table 2 indicates that the VO version only helps at one bandwidth: 4Mbps. For other bandwidth settings, since Q_b does not even change between the two versions, it is not possible to switch to a higher quality without stalls. This happens because FB limits the video quality to only two levels: 720p and 1080p, which incur inflexibility for bitrate selection.

Table 3: Upload region vs. encoded bitrate, stall (S, in seconds), and B2V latency (L, in seconds). Averaged over four 4-minute runs on YouTube, for the *Campus* video.

Upload Region	Encoded Rate (Kbps)	Walking		Driving	
		S	L	S	L
360° ×180°	8297	199	55	170	42
180° ×180°	4823	154	53	135	39
140° ×135°	3667	92	35	34	29
100° ×90°	1997	30	34	0.3	28

Overall, our findings suggest that even realizing a simple viewport-adaptive streaming scheme with perfect knowledge of the user's viewport trajectory is not trivial. Issues such as cautious/inefficient rate adaptation and a limited number of quality levels may diminish the benefits of viewport-adaptiveness. Thus, more research is needed in this direction.

6.2 Adaptive Upload from Broadcaster

On the broadcaster side, ensuring a good upload quality is critical because any disruption on an upload will affect *all* viewers. Recall from §4.1 that neither YT nor FB performs rate adaptation for live 360° video upload. Here instead of investigating traditional rate adaptation in the quality dimension, we consider doing that in the *spatial dimension* – a unique opportunity for 360° live video upload: when the broadcaster-side network condition degrades, the broadcaster adaptively shrinks the view being uploaded from full panorama (360° ×180°) to smaller regions such as 180° ×150°. This is feasible because for many live events such as sports, performance, and ceremony, viewers' "hotspot" areas are oftentimes narrower than the full panorama. Therefore reducing the upload region can considerably reduce the bandwidth utilization while maintaining a good or even better quality of the hotspot area, assuming the hotspot is correctly identified.

We next demonstrate the potential benefits of the above approach when applied to commercial live video streaming infrastructures. We pick the *Campus* video and re-encode it into four versions with the same quality (CRF=27) but different upload regions: 360° ×180° (full panorama), 180° ×180°, 140° ×135°, and 100° ×90° (regular FoV), assuming the hotspot center is (lat=0°, lon=0°). As shown in Table 3, shrinking the upload region significantly reduces the video size by 42% to 76%. The other two videos show similar results.

We then replay these video versions as live YouTube feeds on LIME while throttling the broadcaster-side bandwidth to emulate a mobile broadcaster experiencing fluctuating bandwidth. We pick two cellular uplink bandwidth traces [37] captured from a cellular carrier when the mobile phone is (1) held by a walking pedestrian and (2) mounted on a moving vehicle. Since the original traces' uplink bandwidth is a bit low, we uniformly scale it up by ~1.5× to make the average bandwidth around 4Mbps. Both traces' bandwidth is highly fluctuating, with the standard deviation being 2.1Mbps (walking) and 1.9Mbps (driving). We use Linux `tc` to replay both bandwidth traces at the broadcaster (no throttling on the viewer side), and measure stalls and B2V latency for a controlled player using LIME. As shown in Table 3, shrinking the upload region effectively reduces both the stalls and latency for YouTube. For the

walking (driving) uplink bandwidth profile, the stall duration is reduced by up to 85% (99.9%) and the B2V latency by up to 38% (33%), compared to uploading the panoramic scene.

7 CONCLUDING REMARKS

This paper presents LIME, a generic and flexible measurement system for both commercial and experimental live video streaming platforms. Using LIME, we investigate two popular 360° live video streaming platforms, YouTube and Facebook, through a crowd-sourced measurement study involving 548 paid viewers. We also perform *controlled* experiments – by replacing crowd-sourced viewers with in-lab players – to quantify the potential benefits brought by viewport adaptiveness to commercial live video streaming services.

We learn two high-level lessons from the above efforts. First, LIME is a powerful tool for quickly performing large scale and repeatable live video streaming experiments. Second, even at its current form of performing monolithic streaming, commercial 360° live video streaming systems are fairly complex, and their performance depends on a wide range of factors across all entities (broadcaster, servers, viewers, and the underlying network). Ignoring any factor such as client-side processing overhead may lead to severely degraded QoE.

ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their valuable comments. We also thank Prof. Kevin Almeroth for shepherding our paper. Feng Qian's research was supported in part by NSF Award #1750890 and a Google Faculty Award.

REFERENCES

[1] Amazon Mechanical Turk. https://www.mturk.com.
[2] Equirectangular Projection. https://en.wikipedia.org/wiki/Equirectangular_projection.
[3] HTML Audio/Video DOM Reference. https://www.w3schools.com/tags/ref_av_dom.asp.
[4] Insta360. https://www.insta360.com/.
[5] Mitmproxy. https://mitmproxy.org/.
[6] Next-generation video encoding techniques for 360 video and VR. https://goo.gl/pNArPy.
[7] Open Broadcaster Software. https://obsproject.com/.
[8] Tesseract-OCR. https://github.com/tesseract-ocr/tesseract.
[9] testmy.net – Internet Speed Test. https://testmy.net/.
[10] Under the hood: Building 360 video. https://code.facebook.com/posts/1638767863078802.
[11] Afzal, S., Chen, J., and Ramakrishnan, K. Characterization of 360-degree videos. In *VR/AR Network* (2017).
[12] Almquist, M., Almquist, V., Krishnamoorthi, V., Carlsson, N., and Eager, D. The prefetch aggressiveness tradeof in 360 video streaming. In *Proceedings of ACM Multimedia Systems Conference. Amsterdam, Netherlands* (2018).
[13] Bao, Y., Wu, H., Zhang, T., Ramli, A. A., and Liu, X. Shooting a moving target: Motion-prediction-based transmission for 360-degree videos. In *Big Data (Big Data), 2016 IEEE International Conference on* (2016), IEEE, pp. 1161–1170.
[14] Broeck, M. V. d., Kawsar, F., and Schöning, J. It's all around you: Exploring 360 video viewing experiences on mobile devices. In *Proceedings of the 2017 ACM on Multimedia Conference* (2017), ACM, pp. 762–768.
[15] Corbillon, X., Devlic, A., Simon, G., and Chakareski, J. Optimal set of 360-degree videos for viewport-adaptive streaming. In *Proceedings of the 2017 ACM on Multimedia Conference* (2017), ACM, pp. 943–951.
[16] Corbillon, X., Simon, G., Devlic, A., and Chakareski, J. Viewport-adaptive navigable 360-degree video delivery. In *Communications (ICC), 2017 IEEE International Conference on* (2017), IEEE, pp. 1–7.
[17] Deng, J., Tyson, G., Cuadrado, F., and Uhlig, S. Internet scale user-generated live video streaming: The twitch case. In *PAM* (2017).
[18] Dimopoulos, G., Leontiadis, I., Barlet-Ros, P., Papagiannaki, K., and Steenkiste, P. Identifying the root cause of video streaming issues on mobile devices. In *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies* (2015).
[19] Duanmu, Z., Zeng, K., Ma, K., Rehman, A., and Wang, Z. A Quality-of-Experience Index for Streaming Video. *IEEE Journal of Selected Topics in Signal Processing 11*, 1 (2017), 154–166.
[20] Graf, M., Timmerer, C., and Mueller, C. Towards Bandwidth Efficient Adaptive Streaming of Omnidirectional Video over HTTP: Design, Implementation, and Evaluation. In *MMSys* (2017).
[21] Hei, X., Liang, C., Liang, J., Liu, Y., and Ross, K. W. A measurement study of a large-scale p2p iptv system. *IEEE transactions on multimedia 9*, 8 (2007), 1672–1687.
[22] Huang, T.-Y., Johari, R., McKeown, N., Trunnell, M., and Watson, M. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. *ACM SIGCOMM Computer Communication Review 44*, 4 (2015), 187–198.
[23] Jiang, N., Swaminathan, V., and Wei, S. Power Evaluation of 360 VR Video Streaming on Head Mounted Display Devices. In *MMSys* (2017).
[24] Kaytoue, M., Silva, A., Cerf, L., Meira Jr, W., and Raïssi, C. Watch me playing, i am a professional: a first study on video game live streaming. In *Proceedings of the 21st International Conference on World Wide Web* (2012), ACM, pp. 1181–1188.
[25] Lee, J., Kim, B., Kim, K., Kim, Y., and Noh, J. Rich360: optimized spherical representation from structured panoramic camera arrays. *ACM Transactions on Graphics (TOG) 35*, 4 (2016), 63.
[26] Li, Y., Zhang, Y., and Yuan, R. Measurement and analysis of a large scale commercial mobile internet tv system. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference* (2011), ACM, pp. 209–224.
[27] Liu, X., Xiao, Q., Gopalakrishnan, V., Han, B., Qian, F., and Varvello, M. 360 innovations for panoramic video streaming. In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks* (2017), ACM, pp. 50–56.
[28] Ma, X., Cackett, M., Park, L., Chien, E., and Naaman, M. Web-Based VR Experiments Powered by the Crowd. In *WWW* (2018).
[29] Mao, H., Netravali, R., and Alizadeh, M. Neural Adaptive Video Streaming with Pensieve . In *Proceedings of SIGCOMM 2017* (2017), ACM, pp. 197–210.
[30] Moorthy, A. K., Choi, L. K., Bovik, A. C., and de Veciana, G. Video Quality Assessment on Mobile Devices: Subjective, Behavioral and Objective Studies. *IEEE Journal of Selected Topics in Signal Processing 6*, 6 (2012), 652–671.
[31] Nasrabadi, A. T., Mahzari, A., Beshay, J. D., and Prakash, R. Adaptive 360-degree video streaming using scalable video coding. In *Proceedings of the 2017 ACM on Multimedia Conference* (2017), ACM, pp. 1689–1697.
[32] Petrangeli, S., Swaminathan, V., Hosseini, M., and De Turck, F. An HTTP/2-based adaptive streaming framework for 360° virtual reality videos. In *Proceedings of MM 2017* (2017), ACM, pp. 1–9.
[33] Pires, K., and Simon, G. Youtube live and twitch: a tour of user-generated live streaming systems. In *MMSys* (2015).
[34] Qi, Y., and Dai, M. The Effect of Frame Freezing and Frame Skipping on Video Quality. In *International Conference on Intelligent Information Hiding and Multimedia Signal Processing* (2006).
[35] Qian, F., Han, B., Ji, L., and Gopalakrishnan, V. Optimizing 360 Video Delivery Over Cellular Networks. In *All Things Cellular* (2016).
[36] Qian, F., Han, B., Xiao, Q., and Gopalakrishnan, V. Flare: Practical Viewport-Adaptive 360-Degree Video Streaming for Mobile Devices. In *ACM MobiCom* (2018).
[37] Raca, D., Quinlan, J. J., Zahran, A. H., , and Sreenan, C. J. Beyond Throughput: a 4G LTE Dataset with Channel and Context Metrics. In *MMSys* (2018).
[38] Siekkinen, M., Masala, E., and Kamarainen, T. A First Look at Quality of Mobile Live Streaming Experience: the Case of Periscope. In *IMC* (2016).
[39] Silverston, T., and Fourmaux, O. Measuring P2P IPTV Systems. In *Proceedings of NOSSDAV* (2007).
[40] Sripanidkulchai, K., Maggs, B., and Zhang, H. An analysis of live streaming workloads on the internet. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement* (2004), ACM, pp. 41–54.
[41] Sun, L., Duanmu, F., Liu, Y., Wang, Y., Ye, Y., Shi, H., and Dai, D. Multi-path multi-tier 360-degree video streaming in 5g networks. In *Proceedings of the 9th ACM Multimedia Systems Conference* (2018), ACM, pp. 162–173.
[42] Tang, J. C., Venolia, G., and Inkpen, K. M. 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* (2016), ACM, pp. 4770–4780.
[43] Wang, B., Zhang, X., Wang, G., Zheng, H., and Zhao, B. Y. Anatomy of a Personalized Livestreaming System. In *IMC* (2016).
[44] Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing 13*, 4 (2004).
[45] Xiao, M., Zhou, C., Liu, Y., and Chen, S. Optile: Toward optimal tiling in 360-degree video streaming. In *Proceedings of the 2017 ACM on Multimedia Conference* (2017), ACM, pp. 708–716.
[46] Xie, L., Xu, Z., Ban, Y., Zhang, X., and Guo, Z. 360ProbDASH: Improving QoE of 360 Video Streaming Using Tile-based HTTP Adaptive Streaming. In *Proceedings*

of MM 2017 (2017), ACM, pp. 315–323.

[47] Xie, L., Zhang, X., and Guo, Z. Cls: A cross-user learning based system for improving qoe in 360-degree video adaptive streaming. In *2018 ACM Multimedia Conference on Multimedia Conference* (2018), ACM, pp. 564–572.

[48] Xie, X., and Zhang, X. Poi360: Panoramic mobile video telephony over lte cellular networks. In *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies* (2017), ACM, pp. 336–349.

[49] Yin, X., Jindal, A., Sekar, V., and Sinopoli, B. A control-theoretic approach for dynamic adaptive video streaming over HTTP. In *Proceedings of SIGCOMM 2015* (2015), ACM, pp. 325–338.

[50] Zhang, C., and Liu, J. On crowdsourced interactive live streaming: a twitch. tv-based measurement study. In *NOSSDAV* (2015).

[51] Zhi, L., Anne, A., Ioannis, K., Anush, M., and Megha, M. Toward A Practical Perceptual Video Quality Metric, 2016.

[52] Zhou, C., Li, Z., and Liu, Y. A Measurement Study of Oculus 360 Degree Video Streaming. In *MMSys* (2017).