360° Innovations for Panoramic Video Streaming

Xing Liu[1] Qingyang Xiao[1] Vijay Gopalakrishnan[2] Bo Han[2] Feng Qian[1] Matteo Varvello[2]
[1]Indiana University [2]AT&T Labs – Research

ABSTRACT

360-degree videos are becoming increasingly popular on commercial platforms. In this position paper, we propose a holistic research agenda aiming at improving the performance, resource utilization efficiency, and users' quality of experience (QoE) for 360° video streaming on commodity mobile devices. Based on a Field-of-View (FoV) guided approach that fetches only portions of a scene that users will see, our proposed research includes the following: robust video rate adaptation with incremental chunk upgrading, big-data-assisted head movement prediction and rate adaptation, novel support for multipath streaming, and enhancements to live 360° video broadcast. We also show preliminary results demonstrating promising performance of our proof-of-concept 360° video streaming system on which our proposed research are being prototyped, integrated, and evaluated.

1 INTRODUCTION

The predominance of video streaming in today's Internet shows no sign of weakening. In Q4 2016, mobile videos have eventually surpassed desktop videos in terms of online viewing time [4]. Today's mobile videos are escalating in many dimensions including resolution, frame rate, codec, and in particular, the interaction method (*e.g.,* 360° video and drone-assisted video). Such changes are fueled by multiple factors including faster mobile networks (LTE and 5G), new video types, more powerful devices, and affordable gears such as Virtual Reality (VR) headsets.

In this position paper, we explore several research directions for streaming 360° videos, also known as *immersive* or *panoramic* videos. 360° videos are expected to become "the next frontier in mobile video" [8]. As a critical component of VR, they provide users with an immersive viewing experience that far exceeds what regular videos can offer. They are becoming increasingly popular on commercial platforms such as YouTube with the top videos being viewed more than 60 million times.

Maintaining good Quality of Experience (QoE) for 360° videos over bandwidth-limited links on commodity mobile devices remains challenging. First, 360° videos are *large*: under the same perceived quality, 360° videos have around 5x larger sizes than conventional videos. Second, 360° videos are *complex*: sophisticated projection and content representation schemes may incur high overhead. For example, the projection algorithm used by Oculus 360 requires servers to maintain up to 88 versions of the same video [46] (see §2). Third, 360° videos are still *under-explored*: there is a lack of real-world experimental studies of key aspects such as rate adaptation, QoE metrics, and cross-layer interactions (*e.g.,* with TCP and web protocols such as HTTP/2).

The contribution of this paper is a research agenda consisting of several novel ideas that we envision will become building blocks of next-generation 360° video systems:

- We employ a Field-of-View (FoV) guided approach that fetches only portions of a scene users will see. We pinpoint a fundamental mismatch between today's popular encoding schemes (*e.g.,* H.264/AVC and H.265/HEVC) and FoV-guided 360° streaming: these schemes lack the capability of *incrementally upgrading* a fetched portion to a higher quality. We thus propose a rate adaptation scheme with a "delta encoding" design that can substantially improve the adaptiveness in face of FoV prediction errors (§3.1).
- We use *big data analytics* to facilitate accurate head movement prediction (HMP), a key prerequisite for FoV-guided streaming. Specifically, we propose to jointly consider several dimensions including (1) viewing statistics of the same video across users, (2) viewing behaviors over multiple videos of a single user, and (3) other contextual information such as users' poses and engagement levels. Such a "data fusion" approach provides key intelligence for FoV-guided prefetching (§3.2).
- We propose enhancements that allow 360° videos to be efficiently streamed over multiple network paths such as WiFi and cellular. The basic idea is to leverage the *spatial and temporal priorities* of 360° video chunks to strategically assign each chunk to an appropriate network path. Compared to application-agnostic approaches such as MultiPath TCP (MPTCP) [5], our content-aware multipath scheme brings better performance and more flexibility (§3.3).
- We propose ideas for improving the performance of live 360° videos, such as broadcaster-side optimizations and realtime crowd-sourcing for HMP prediction. These ideas are motivated by a preliminary characterization study of today's commercial live 360° video broadcast platforms: Facebook, YouTube, and Periscope (§3.4).

We are currently working on realizing the above ideas and integrating them into a holistic 360° streaming system on commercial off-the-shelf (COTS) mobile devices with various system-level optimizations as sketched in §3.5. We describe related work in §2 and conclude the paper in §4.

HotNets-XVI, November 30-December 1, 2017, Palo Alto, CA, USA

ACM ISBN 978-1-4503-5569-8/17/11…$15.00
https://doi.org/10.1145/3152434.3152443

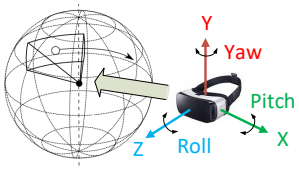

Figure 1: Watching 360-degree videos.

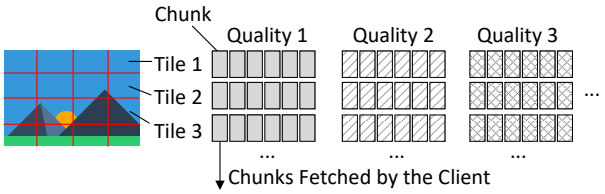

Figure 2: Server-side content organization in Sperke.

2 BACKGROUND AND RELATED WORK

Background. 360° videos are recorded by omnidirectional cameras which capture all 360 degrees of a scene that can be "wrapped" onto a 3D sphere. There are COTS portable omnidirectional cameras that are either standalone (*e.g.,* Acer Holo 360° with built-in LTE [2]) or attachable to a smartphone (*e.g.,* Insta360 [3]). Users can watch 360° videos directly on commodity smartphones or through affordable VR devices such as a Google Cardboard [1]. When watching a 360° video, a viewer at the center of the sphere can freely control her viewing direction by changing the pitch, yaw, and roll as shown in Figure 1. During a playback, in addition to performing regular video decoding, the player also needs to apply projection to render the content in the user's current FoV based on her head orientation. There are different projection algorithms such as equirectangular projection [11] used by YouTube and Cube Map [10] employed by Facebook. The width and height of the FoV are usually fixed parameters of a VR headset.

Related Work. Today's major 360° video providers (*e.g.,* YouTube and Facebook) employ *FoV-agnostic* streaming that always delivers the entire panoramic view, leading to tremendous bandwidth waste [16, 37]. To play HD 360° videos smoothly over networks with limited bandwidth, we can employ *FoV-guided* streaming that focuses on providing high-quality views within users' FoV. Below, we describe two main FoV-guided approaches: *tiling* and *versioning*.

Tiling. The 360° video is spatially segmented into *tiles*. Only tiles within the FoV are downloaded at high quality, whereas the remaining tiles are delivered at lower qualities or not delivered at all [16, 23, 24, 27, 37]. Prior studies demonstrated via trace-driven simulations that tiling provides significant bandwidth saving (typically 45% [16] and 60% to 80% [37]) compared to the FoV-agnostic approach. Tiling imposes minimal load at the server while increasing the load at the client, which needs to determine the set of tiles to fetch and then "stitch" them together. We show in §3.5 that it is feasible to realize this approach on COTS smartphones.

Versioning. The 360° video is encoded into multiple versions each having a different high-quality region; the player needs to pick the appropriate version based on user's viewing direction [18, 46]. This approach simplifies the fetching, decoding, and rendering logic at the client's player, but incurs substantial overhead at the server that needs to maintain a large number of versions of the same video (*e.g.,* up to 88 for Oculus 360 [46]).

In the multimedia community, a plethora of work focused on panoramic video generation from either a single camera or multiple cameras [21, 41, 43]. Researchers also proposed different projection and encoding schemes [6, 10, 27, 32, 33, 39]. Some other studies investigated other aspects such as measurement [13, 46], energy consumption [30], and optimization through edge computation [34]. It is worth mentioning that the vast majority of recent mobile 360° video research [16–20, 27, 30, 37, 45] is based on *simulation or trace-driven emulation*. To the best of our knowledge, only [24] used a proprietary system implementation that is in early-stage, *e.g.,* lacking key components such as rate adaptation and head movement prediction. This is in sharp contrast with conventional video streaming research where many mature and open-source players and tools are available.

3 THE RESEARCH AGENDA

We have recently started developing Sperke, a FoV-guided adaptive mobile 360° video streaming framework. Sperke employs a *tiling-based* approach to avoid storing too many video versions at the server side. Sperke has three goals: (1) smooth playback based on robust head movement prediction (HMP), (2) interchangeable projection, rate adaptation, and HMP algorithms as the ones we propose in this paper, and (3) compatibility with COTS devices. As shown in Figure 2, Sperke follows the DASH [38] paradigm and encodes a panoramic video into multiple qualities; each quality is *spatially* segmented into multiple *tiles*, which are then *temporally* split into *chunks*. A chunk $C(q, l, t)$ is thus the smallest downloadable unit in Sperke where q, l, and t are the quality level, tile ID, and chunk starting time, respectively. All chunks have the same duration (*e.g.,* one or two seconds), and are fetched by the client's player based on estimated network conditions and HMP. Chunks are then decoded, projected, and rendered according to user's current FoV (details in §3.5).

We will use Sperke as an "infrastructural base" upon which our proposed research will be prototyped and evaluated. We next describe our research agenda that adds several salient features and optimizations to the Sperke framework to improve its performance, resource efficiency, and usability.

3.1 Video Rate Adaptation with Incremental Chunk Upgrades

3.1.1 Incremental Chunk Upgrading Support. Today's 360° and non-360° videos share the same set of encoding schemes such as H.264/AVC, H.265/HEVC, VP9, *etc.* We make a key observation that *there is a fundamental mismatch between them and FoV-guided streaming* due to HMP's imperfection. To see the reason, assume that the HMP algorithm predicts that the user will look at FoV X. Due to the

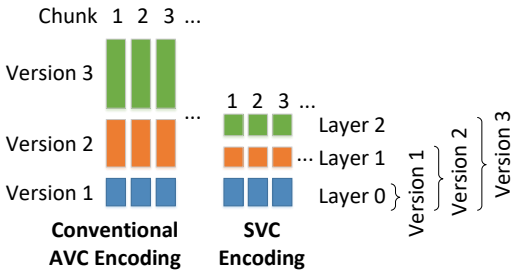

Figure 3: AVC vs. SVC encoding.

inherent human "randomness", the user oftentimes ends up looking at FoV $X' \neq X$. Prior studies indicate that such prediction errors $|X - X'|$ are usually small and can be corrected over time [16, 37]. To tolerate them, the player needs to fetch more tiles surrounding the predicted FoV area X. Such tiles are called "out-of-sight (OOS)" tiles [37] because they will not be visible if the HMP is perfect. To save bandwidth, OOS tiles are downloaded in lower qualities. For example, the further away they are from X, the lower their qualities will be. As a result, some chunks in X' will be displayed at lower qualities unless they are discarded and replaced by higher-quality chunks, as shown on the left side of Figure 3. Ideally, we would need an *incremental chunk upgrade* mechanism: if the HMP discovers some downloaded OOS chunks will be displayed, the player can upgrade them to higher qualities by only downloading the delta.

Such incremental chunk upgrades can be realized by Scalable Video Encoding (SVC), which was standardized as an H.264 extension [12], and will also be available in H.265 [40]. The basic idea of SVC is to encode a chunk into ordered *layers*: one *base layer* (Layer 0) with the lowest playable quality, and multiple *enhancement layers* (Layer i >0) that further improve the chunk quality based on layer $(i-1)$. When playing a chunk at layer $i > 0$, the player must have all its layers from 0 to i, as shown on the right side of Figure 3. As of today, SVC has registered little commercial deployment likely due to its complexity and a lack of use cases [31]. But we envision that SVC will find its unique application in 360° video streaming and VR.

We are unaware of any prior effort that applies SVC to 360° videos or VR. The closest approach to this idea is a two-layer encoding scheme proposed in [19]. Our approach is different though since we allow multiple layers and make multi-layer incremental chunk upgrade a *fundamental building block* for tiling-based 360° video streaming.

We plan to add SVC support to Sperke and evaluate its effectiveness. A practical issue is that despite the availability of open-source SVC codecs, today's mainstream smartphones do not have built-in hardware SVC decoders. We thus plan to adopt a cloudlet-based offloading approach in our study: use a local cloudlet [25] to perform runtime transcoding from SVC to AVC, which can then be decoded by mobile GPU.

3.1.2 Video Rate Adaptation (VRA) Algorithm. Now let us consider Video Rate Adaptation (VRA), whose goal is to determine the quality level of each chunk to be fetched so to maximize the user QoE [14] (fewer stalls/skips[1], higher bitrate, and fewer quality changes). Designing a VRA algorithm for tiling-based 360° videos is challenging, because the algorithm needs to make judicious decisions at not only the *quality domain* (which quality level to fetch) but also at the *spatial domain* (which tiles to fetch) based on *multiple input sources* including network bandwidth estimation and HMP, both possibly inaccurate. To address this challenge, we decompose our design into three parts with increasing complexity as to be detailed next.

In the first part, let us assume that the HMP is perfect. Then the FoV-guided 360° VRA essentially falls back to regular (non-360°) VRA. This is because we can generate a sequence of *super chunks* where each super chunk consists of the minimum number of chunks that fully cover the corresponding FoV which we precisely know. We can then apply a regular VRA to the super chunks. A constraint that reduces the complexity is that all chunks within a super chunk will have the same quality (otherwise different subareas in a FoV will have different qualities, thus worsening the QoE). The challenge here is the VRA algorithm design, which can be possibly done by customizing an existing VRA algorithm. There exists a large body of VRA for regular videos such as throughput-based [29], buffer-based [28], and control-theoretic approaches [44]. However some of them may poorly interact with FoV-guided 360° video streaming. For example, buffer-based VRA (*e.g.,* BBA [28]) may not be a good candidate because the HMP prediction window is usually short (§3.2) and may thus limit the video buffer occupancy.

In the second part, we handle inaccurate HMP by strategically *adding OOS chunks* to each super chunk. The number of OOS chunks and their qualities depend on three factors. First, the bandwidth constraint that can either be the physical bandwidth limit or the bandwidth budget configured by the user. Second, the HMP accuracy; ideally, the lower the accuracy is, the more OOS chunks at higher qualities are needed due to the increased randomness; in the worst case when the head movement is completely random, OOS chunks may spread to the entire panoramic scene. Third, data-driven optimizations to further intelligently modify the OOS tiles (details in §3.2). We will develop an OOS chunk selection algorithm that considers the above factors and balances their incurred tradeoffs.

In the third part, we consider *incremental chunk upgrades* (§3.1.1). At runtime, the VRA algorithm continuously updates the chunk scheduling based on the bandwidth prediction and HMP. An updated scheduling may trigger chunks' incremental update (*i.e.,* fetching enhancement layers). Two decisions need to be carefully made: (1) *upgrade or not*: upgrading improves the quality while not upgrading saves bandwidth for fetching future chunks; (2) *when to upgrade*: upgrading too

[1]For realtime (live) streaming, chunks not received by their deadlines are skipped. For non-realtime streaming, if a chunk cannot be fetched by its deadline, typically a stall (rebuffering) will occur.

early may lead to extra bandwidth waste since the HMP may possibly change again in the near future, while upgrading too late may miss the playback deadline.

We will integrate the above three parts into a holistic 360° VRA algorithm. Note that our proposed scheme can be further extended to support a *hybrid SVC/AVC scheme*: the server prepares both the SVC and AVC versions of a 360° video; for chunks whose qualities are not likely to upgrade, we can directly fetch their AVC versions to avoid the (reasonable yet not negligible) performance/bandwidth overhead of SVC.

3.2 Big Data Analytics for HMP and VRA

As a critical component of FoV-guided video streaming, HMP improves the user experience by allowing the player to prefetch chunks that the user is about to see. Previous studies involving small-scale user trials [16, 37] indicate that for many videos, *HMP at a short time scale* (*e.g.,* hundreds of milliseconds to up to two seconds) with a reasonable accuracy can be achieved by learning past head movement readings. This research task instead goes beyond existing in-lab, short-time, small-scale, single-feature studies by investigating how mobile users interact with 360° videos "in the wild". We plan to take *a long-term, crowd-sourced, multi-feature study at Internet scale*, with the goal of facilitating HMP and rate adaptation by learning from big data. Specifically, we will develop a 360° video player app and publish it to mobile app stores such as Google Play. Under users' consent, the app will collect a wide range of information such as (1) the video URL, (2) users' head movement during 360° video playback, (3) user's rating of the video, (4) lightweight contextual information such as indoor/outdoor, watching mode (bare smartphone vs. headset), mobility (stationary vs. mobile), pose (sitting, standing, lying *etc.*). Since we do not host the actual video content and only collect light information (uncompressed head movement data at 50 Hz is less than 5 Kbps), our system can easily scale. To give users incentives, the app will contain some attractive features such as recommending popular 360° videos from multiple sources. The IRB application of this study is currently in progress.

We will then analyze such "big data" and use it to either directly improve HMP or help VRA (§3.1.2). Specifically, we want to answer three research questions.

First, how to leverage *multiple* users' viewing statistics of the *same* video to guide chunk fetching. Intuitively, we can give "popular" chunks (viewed by most users) higher priorities when prefetching them, thus making *long-term prediction* feasible. Theoretically, when being integrated with VRA, this can be formulated as a stochastic optimization problem: using chunks' viewing probabilities to optimally find the chunks to download (as well as their qualities) such that the QoE is maximized.

Second, how to mine the *same* user's viewing behaviors in the long run over *multiple* videos to customize the tile fetching strategy for that particular user. For example, a user's head movement speed can be learned to bound the latency requirement for fetching a distant tile (*e.g.,* elderly people tend to move their heads slower than teenagers).

Priority	Spatial	Temporal
High	FoV chunks	urgent chunks
Low	OOS chunks	regular chunks

Table 1: Spatial & temporal priorities in 360° videos.

Third, how to utilize the *contextual information* to further facilitate HMP. For example, when the user is lying on a couch or bed, it is quite difficult for her to view a direction that is 180° behind. In another example, we can leverage eye gaze tracking to analyze the user's engagement level [15], which possibly indicates the likelihood of sharp hard movement (this will be studied in a lab setting).

Finally, the above features will be *jointly* exploited and integrated with the HMP and VRA schemes. For example, we can use the crowd-sourced data to *add* OOS chunks, and use the user's head movement speed bound and the contextual information to *prune* OOS chunks.

3.3 Multipath Streaming Support

Multiple network interfaces (*e.g.,* WiFi and LTE) are widely supported by today's mobile devices. Multipath transport allows applications to use multiple interfaces simultaneously. It provides new opportunities for drastically improving the mobile app performance, including video streaming [26]. The state-of-the-art multipath solution is MPTCP [5]. It uses a content-agnostic multipath model: the upper-layer video server application regards all available paths as a single logical path, while the multipath scheduler transparently splits the video bitstream over the actual paths. Such a paradigm is simple and general. Also it largely makes sense for a conventional non-360° video that consists of only a single stream.

Although vanilla MPTCP can be directly applied to 360° video streaming, we identify opportunities for further improvement. Our key observation is that tiling-based 360° videos can be naturally broken down into "substreams", each with different priorities. We identify two types of priorities listed in Table 1. For *spatial priority*, FoV chunks are more important than OOS chunks (§3.1.1); for *temporal priority*, "urgent chunks" are chunks that have a very short playback deadline due to, for example, a correction of a previous inaccurate HMP. To avoid imminent stalls, they should be assigned a higher priority than regular (non-urgent) chunks.

Motivated by the above, we propose to go beyond the "bitstream splitting" model of MPTCP by strategically *using application knowledge to guide multipath decisions*, in order to achieve better video QoE. We plan to leverage spatial and temporal priorities in the context of 360° video streaming. Consider the spatial priority as an example. Assume there are two paths with different qualities (loss rate, latency, *etc.*). One possible approach is to prioritize FoV and OOS chunks over the high-quality and low-quality paths, respectively, and to deliver them in different transport-layer QoS (reliable vs. best-effort respectively). Doing so offers *application-layer benefits* in that it facilitates faster delivery of FoV chunks

by sacrificing the quality/performance of OOS chunks – a desirable tradeoff in many cases. Multipath also provides *transport-layer benefits*: since all paths are now fully decoupled, issues such as cross-path out-of-order will be eliminated, leading to significant performance improvement [36].

Besides developing multipath strategies for 360° videos on which no prior effort exists, we also plan to apply the general concept of *application-assisted multipath optimization*, a philosophy differing from MPTCP's content-agnostic principle, to other multimedia applications. We will also explore how other transport-layer primitives such as network coding [22] can be leveraged in this context.

3.4 Improving Live 360° Video Broadcast

So far our discussions focused on on-demand 360° videos. We now consider live 360° video broadcast, where a *broadcaster* device uploads the recorded panoramic live video to a server which then disseminates it to multiple *viewers*.

3.4.1 Pilot Characterization Study. Fueled by today's high-speed access networks and affordable panoramic cameras, live 360° video broadcast started to gain popularity on commercial platforms. Since we are unaware of any prior study, we conduct in-lab measurements of live 360° videos on three popular platforms: Facebook, YouTube, and Periscope. The goal of our measurements is to understand various properties of live 360° video broadcast, such as the protocol usage, rate adaptation, and performance. In our experimental testbed, we use an Insta360 panoramic camera [3] attached to a Google Nexus 5X smartphone to broadcast 360° videos. The receivers consist of several laptops where viewers watch the video lively, for example, on the broadcaster's Facebook timeline. By default both the broadcaster and the viewers use high-speed WiFi networks. We leveraged several tools including tcpdump, mitmproxy [9] (for HTTPS inspection), and tc (for emulating poor network conditions).

Protocols. 360° and non-360° live video [35] broadcast share a similar architecture: a server disseminates the live content uploaded by the broadcaster to all viewers. For 360° live broadcast on the three platforms, we find that the *upload* path (from the broadcaster to the server) uses RTMP (Real-Time Messaging Protocol), a proprietary live streaming protocol over TCP[2]. The *download* path (from the server to viewers) employs either regular pull-based HTTP DASH (Facebook and YouTube) or push-based RTMP (Periscope).

FoV-guided Streaming. We find that neither Facebook nor YouTube employs FoV-guided streaming. Henceforth the broadcaster has always to upload full panoramic views, which are then entirely delivered to the viewers, possibly in lower qualities due to rate adaptation (see below). As a result, under the same perceived quality, 360° live videos are about 4 to 5 times larger than non-360° live videos. Periscope likely does *not* use FoV-guided streaming either, but we are not able to confirm that because its RTMP traffic is encrypted.

[2]RTMP was developed by Adobe, which has released an incomplete protocol specification for public use [7].

Upload BW	Download BW	End-to-End Latency (second)		
		Facebook	Periscope	YouTube
No limit	No limit	9.2	12.4	22.2
2Mbps	No limit	11	22.3	22.3
No limit	2Mbps	9.3	20	22.2
0.5Mbps	No limit	22.2	53.4	31.5
No limit	0.5Mbps	45.4	61.8	38.6

Table 2: E2E latency under different network conditions.

Rate Adaptation. Regardless of the video platform, our experiments show that no rate adaptation is currently used during a live 360° video upload. Instead, video quality is either fixed or manually specified by the broadcaster.

With respect to download, Facebook and YouTube employ DASH-style rate adaptation. Specifically, the viewer periodically requests for a Media Presentation Description (MPD) file that contains the meta data (*e.g.,* URL, quality, codec info) for recently generated video chunks that are (re)encoded by the server into multiple quality levels (720p/1080p for Facebook and six levels from 144p to 1080p for YouTube). The viewer will then select an appropriate quality level for each chunk and fetch it over HTTPS.

End-to-End (E2E) Latency. We call E2E latency the elapsed time between when a real-world scene appears and its viewer-side playback time. This latency consists of delays incurred at various components including network transmission, video encoding, and buffering at the three entities (broadcaster, server, and viewer). E2E latency is a key QoE metric for live video streaming [42], which we measure as follows. The broadcaster points its camera to a digital clock that displays time T_1; moments later when this scene is seen by the viewer, the clock displays time T_2. We use a separate camera to simultaneously record T_1 (on the viewer's screen) and T_2 to compute the E2E latency as $T_2 - T_1$.

Table 2 shows a summary of the E2E latency measured for Facebook, Periscope, and YouTube under different network conditions. Each value in the table is the average E2E latency computed across 3 experiments (stddev/mean ranges from 1% to 17%); the camera recording quality is fixed at 1080p. We make two observations. First, the "base" latency when the network bandwidth is not limited is non-trivial. Second, as the network condition worsens, we observe degraded video quality exhibiting stall and frame skips (not shown in Table 2) as well as inflated E2E latency, particularly when the bandwidth is reduced to 0.5Mbps.

3.4.2 Improving the state-of-the-art. Despite being preliminary, our measurements indicate that live 360° video broadcast can be improved in various aspects. We next describe two directions we plan to pursue.

First, we will design a VRA scheme for live 360° video upload using lessons learned from non-live 360° VRA (§3.1.2). Here a novel design aspect is that, when the network quality at the broadcaster side degrades, instead of stalling/skipping frames or decreasing the quality of the panoramic view, the broadcaster can have an additional option of what we call *spatial fall-back* that adaptively reduces the overall "horizon"

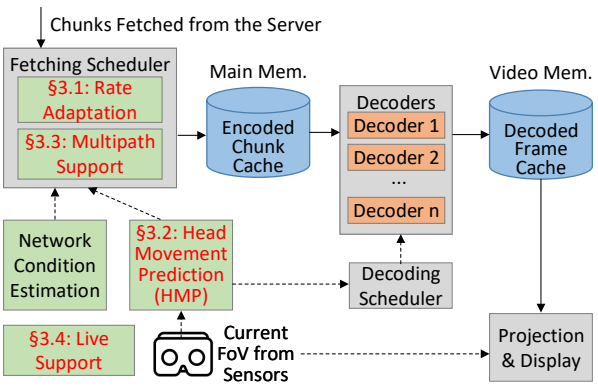

Figure 4: Client-side logic of Sperke**.**

being uploaded (*e.g.,* from 360° to 180°) in order to reduce the bandwidth consumption. The intuition is that for many live broadcasting events such as sports, performance, ceremony, *etc.*, the "horizon of interest" is oftentimes narrower than full 360°. Therefore reducing the uploaded horizon may bring better user experience compared to blindly reducing the quality. The challenge here is to solve the open problem of determining the (reduced) horizon's center and the lower bound of its span (*e.g.,* ideally it should be "wider" than the concert's stage). We envision that this can be achieved by combining several approaches including manual hints from the broadcaster, crowd-sourced HMP prediction (see below), and real-time video content analysis (offloadable to the cloud).

The second research direction is crowd-sourced HMP prediction for live 360° videos at the viewer side. Due to its realtime nature, this task is more challenging than the HMP prediction for non-live 360° videos. Our basic idea is to leverage the fact that different viewers experience different viewing latency (§3.4.1): when many viewers are present, due to the heterogeneity of their network quality which, together with other factors, dictates the buffering level, the E2E latency across users will likely exhibit high variance as exemplified in Table 2 (this has also been observed in non-360° live streaming measurements [42]). We can therefore use the realtime head movement statistics of low-latency users (whose network qualities are typically good) to help HMP for high-latency users who experience challenging network conditions and thus can benefit from FoV-guided streaming.

We will conduct research on both directions above and integrate our solutions with the Sperke framework. It is also worth mentioning that 360° live broadcast can also benefit from our previously introduced primitives such as incremental chunk upgrading (§3.1.1) and application-assisted multipath (§3.3). For example, if the broadcaster employs SVC encoding, then there is no need for the server to perform re-encoding because the client player can directly assemble individual layers into chunks with different qualities.

3.5 System-level Integration & Optimization

We plan to integrate the components described from §3.1 to §3.4 into Sperke, making it a unified streaming platform for both non-live and live 360° videos. This faces numerous

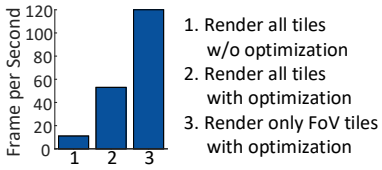

Figure 5: Preliminary performance of the Sperke **player.**

system challenges, the most critical one being dealing with the limited computation and energy resources on the client side (mobile device) where most Sperke's logic resides.

Figure 4 sketches the high-level system architecture we envisage for the client-side Sperke logic. We highlight two system-level optimizations. First, Sperke effectively utilizes multiple hardware decoders on mobile devices (*e.g.,* 8 H.264 decoders for Samsung Galaxy S5 and 16 for Samsung Galaxy S7) *in parallel* to improve the performance. We plan to design a decoding scheduler that assigns encoded chunks to decoders based on their playback time and HMP. Second, we introduce a *decoded chunk cache* (implemented using OpenGL ES Framebuffer Objects) that stores *uncompressed video chunks* in the video memory. Doing so allows decoders to work asynchronously, leading to a higher frame rate. More importantly, when a previous HMP is inaccurate, the cache allows a FoV to be quickly shifted by only changing the "delta" tiles without re-decoding the entire FoV.

Our preliminary measurements indicate that the above optimizations are highly effective by *engineering the low-level streaming pipeline* and *judiciously manipulating raw video data.* As shown in Figure 5, on SGS 7, using a 2K video, 2x4 tiles, and 8 parallel H.264 decoders, the FPS increases from 11 (before optimization) to 53 (after optimization); rendering tiles only in FoV further boosts the FPS to 120. Our approach also significantly outperforms the built-in "tiles" mechanism introduced in the latest H.265 codec [40], which is not designed specifically for 360° videos. Overall, our preliminary system and its measurement results demonstrate the feasibility of building a tiling-based FoV-guided streaming scheme on COTS smartphones using conventional decoders.

4 CONCLUDING REMARKS

Our proposed research optimizes 360° video streaming using interdisciplinary approaches: we creatively apply existing techniques developed from the video coding community in new contexts (§3.1); we use big data and crowd-sourcing to add intelligence to streaming algorithms (§3.2); we utilize cross-layer knowledge to facilitate content distribution over multiple networks (§3.3); we provide new insights in optimizing live 360° video streaming from both broadcaster and viewer sides (§3.4). Finally we will integrate our innovations into Sperke, a holistic system with various system-level optimizations (§3.5).

ACKNOWLEDGEMENTS

We would like to thank the HotNets reviewers for their valuable feedback. The research of Feng Qian was supported in part by a Google Faculty Award.

REFERENCES

[1] Google Cardboard. https://vr.google.com/cardboard/index.html.

[2] Holo 360-degree Camera with Integrated 4G LTE. http://www.anandtech.com/show/11303/acer-announces-holo-360-camera-with-4g-lte.

[3] Insta360 panoramic camera. https://www.insta360.com/.

[4] Mobile vs Desktop Video: How do they compare? https://www.linkedin.com/pulse/mobile-vs-desktop-video-how-do-compare-freddie-benjamin.

[5] Multipath TCP in the Linux Kernel. http://www.multipath-tcp.org.

[6] Next-generation video encoding techniques for 360 video and VR. https://code.facebook.com/posts/1126354007399553/next-generation-video-encoding-techniques-for-360-video-and-vr/.

[7] Real-Time Messaging Protocol (RTMP) specification. http://www.adobe.com/devnet/rtmp.html.

[8] The future of mobile video is virtual reality. https://techcrunch.com/2016/08/30/the-future-of-mobile-video-is-virtual-reality/.

[9] The mitmproxy. https://mitmproxy.org/.

[10] Under the hood: Building 360 video. https://code.facebook.com/posts/1638767863078802.

[11] YouTube live in 360 degrees encoder settings. https://support.google.com/youtube/answer/6396222.

[12] Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEGg, 24th Meeting. http://ip.hhi.de/imagecom_G1/savce/downloads/H.264-MPEG4-AVC-Version8-FinalDraft.pdf, Jun-Jul 2007.

[13] S. Afzal, J. Chen, and K. Ramakrishnan. Characterization of 360-degree videos. In *Proceedings of the Workshop on Virtual Reality and Augmented Reality Network*, pages 1–6. ACM, 2017.

[14] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang. Developing a Predictive Model of Quality of Experience for Internet Video. In *SIGCOMM*, 2013.

[15] X. Bao, S. Fan, A. Varshavsky, K. Li, and R. Roy Choudhury. Your reactions suggest you liked the movie: Automatic content rating via reaction sensing. In *UbiComp 2013*, pages 197–206. ACM, 2013.

[16] Y. Bao, H. Wu, T. Zhang, A. A. Ramli, and X. Liu. Shooting a moving target: Motion-prediction-based transmission for 360-degree videos. In *Big Data (Big Data), 2016 IEEE International Conference on*, pages 1161–1170. IEEE, 2016.

[17] Y. Bao, T. Zhang, A. Pande, H. Wu, and X. Liu. Motion-prediction-based multicast for 360-degree video transmissions. In *SECON 2017*, pages 1–9. IEEE, 2017.

[18] X. Corbillon, A. Devlic, G. Simon, and J. Chakareski. Viewport-adaptive navigable 360-degree video delivery. *arXiv preprint arXiv:1609.08042*, 2016.

[19] F. Duanmu, E. Kurdoglu, S. A. Hosseini, Y. Liu, and Y. Wang. Prioritized buffer control in two-tier 360 video streaming. In *Proceedings of the Workshop on Virtual Reality and Augmented Reality Network*, pages 13–18. ACM, 2017.

[20] 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 *MMSys*, 2017.

[21] C. Fehn, C. Weissig, I. Feldmann, M. Muller, P. Eisert, P. Kauff, and H. BloB. Creation of high-resolution video panoramas of sport events. In *Multimedia, 2006. ISM'06. Eighth IEEE International Symposium on*, pages 291–298. IEEE, 2006.

[22] T. Flach, N. Dukkipati, A. Terzis, B. Raghavan, N. Cardwell, Y. Cheng, A. Jain, S. Hao, E. Katz-Bassett, and R. Govindan. Reducing web latency: the virtue of gentle aggression. *ACM SIGCOMM Computer Communication Review*, 43(4):159–170, 2013.

[23] V. R. Gaddam, M. Riegler, R. Eg, C. Griwodz, and P. Halvorsen. Tiling in interactive panoramic video: Approaches and evaluation. *IEEE Transactions on Multimedia*, 18(9):1819–1831, 2016.

[24] M. Graf, C. Timmerer, and C. Mueller. Towards Bandwidth Efficient Adaptive Streaming of Omnidirectional Video over HTTP: Design, Implementation, and Evaluation. In *MMSys*, 2017.

[25] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan. Towards wearable cognitive assistance. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*, pages 68–81. ACM, 2014.

[26] B. Han, F. Qian, L. Ji, and V. Gopalakrishnan. MP-DASH: Adaptive Video Streaming Over Preference-Aware Multipath. In *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*, pages 129–143. ACM, 2016.

[27] M. Hosseini and V. Swaminathan. Adaptive 360 vr video streaming: Divide and conquer! *arXiv preprint arXiv:1609.08729*, 2016.

[28] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. *ACM SIGCOMM Computer Communication Review*, 44(4):187–198, 2015.

[29] J. Jiang, V. Sekar, and H. Zhang. Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, pages 97–108. ACM, 2012.

[30] N. Jiang, V. Swaminathan, and S. Wei. Power Evaluation of 360 VR Video Streaming on Head Mounted Display Devices. In *MMSys*, 2017.

[31] C. Kreuzberger, D. Posch, and H. Hellwagner. A Scalable Video Coding Dataset and Toolchain for Dynamic Adaptive Streaming over HTTP. *MMSys*, 2015.

[32] J. Lee, B. Kim, K. Kim, Y. Kim, and J. Noh. Rich360: optimized spherical representation from structured panoramic camera arrays. *ACM Transactions on Graphics (TOG)*, 35(4):63, 2016.

[33] J. Li, Z. Wen, S. Li, Y. Zhao, B. Guo, and J. Wen. Novel tile segmentation scheme for omnidirectional video. In *Image Processing (ICIP), 2016 IEEE International Conference on*, pages 370–374. IEEE, 2016.

[34] S. Mangiante, G. Klas, A. Navon, Z. GuanHua, J. Ran, and M. D. Silva. Vr is on the edge: How to deliver 360 videos in mobile networks. In *Proceedings of the Workshop on Virtual Reality and Augmented Reality Network*, pages 30–35. ACM, 2017.

[35] M. K. Mukerjee, D. Naylor, J. Jiang, D. Han, S. Seshan, and H. Zhang. Practical, Real-time Centralized Control for CDN-based Live Video Delivery. In *SIGCOMM*, 2015.

[36] A. Nikravesh, Y. Guo, F. Qian, Z. M. Mao, and S. Sen. An in-depth understanding of multipath tcp on mobile devices: measurement and system design. In *MobiCom 2016*, pages 189–201. ACM, 2016.

[37] F. Qian, L. Ji, B. Han, and V. Gopalakrishnan. Optimizing 360 video delivery over cellular networks. In *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*, pages 1–6. ACM, 2016.

[38] I. Sodagar. The MPEG-DASH standard for multimedia streaming over the Internet. *IEEE MultiMedia*, 18(4):62–67, 2011.

[39] K. K. Sreedhar, A. Aminlou, M. M. Hannuksela, and M. Gabbouj. Viewport-adaptive encoding and streaming of 360-degree video for virtual reality applications. In *Multimedia (ISM), 2016 IEEE International Symposium on*, pages 583–586. IEEE, 2016.

[40] G. J. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand. Overview of the high efficiency video coding (HEVC) standard. *IEEE Transactions on circuits and systems for video technology*, 22(12):1649–1668, 2012.

[41] R. Szeliski. Image Alignment and Stitching: A Tutorial. Technical Report MSR-TR-2004-92, Microsoft Research, 2004.

[42] B. Wang, X. Zhang, G. Wang, H. Zheng, and B. Y. Zhao. Anatomy of a personalized livestreaming system. In *Proceedings of the 2016 ACM on Internet Measurement Conference*, pages 485–498. ACM, 2016.

[43] W. Xu and J. Mulligan. Panoramic video stitching from commodity HDTV cameras. *Multimedia systems*, 19(5):407–426, 2013.

[44] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli. A control-theoretic approach for dynamic adaptive video streaming over http. *ACM SIGCOMM Computer Communication Review*, 45(4):325–338, 2015.

[45] 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 2016 ACM on Multimedia Conference*, pages 601–605. ACM, 2016.

[46] C. Zhou, Z. Li, and Y. Liu. A measurement study of oculus 360 degree video streaming. *MMSys*, 2017.