When 360-degree Video Meets 5G: A Measurement Study of Delay in Live Streaming

Haitao Liu[1,2], Xinyi Zhang[1], Xiang Zhou[3], Tianyun Zhang[1],
Yanbiao Li[1,2], Zhenyu Li[2,4], Gaogang Xie[1,2*]
[1]Computer Network Information Center, Chinese Academy of Sciences, Beijing, China
[2]University of Chinese Academy of Sciences, Beijing, China
[3]Academy of Broadcasting Science, NRTA, Beijing, China
[4]Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China

ABSTRACT

Live 360° video can provide users with an immersive viewing and interactive experience, and end-to-end live delay is an important metric in quality of experience (QoE). The low latency and high bandwidth nature makes 5G become promising to deliver huge volume of data in 360° video live streaming. However, most of the existing works measured 360° video live delay under non-5G access (e.g., WiFi, LTE or Ethernet). The delay measurement study under 5G access is not comprehensive enough yet. To further validate the feasibility of delivering 360° video live streaming under 5G access, in this paper, we firstly build our 360° video live streaming system based on a series of open-source projects, then measure end-to-end live delay, computation and communication delay of/between each component (i.e., camera, server and client). We find that 5G does behave more superior compared with WiFi and Ethernet access, benefiting from its extremely low downstream delay although its upstream delay is much higher. In addition, we also find that the major delay bottleneck locates in the computation of each stage rather than the communication between them. To further reduce the end-to-end live delay, it's necessary to improve the hardware processing capability.

CCS CONCEPTS

• **Networks → Network measurement**; • **Information systems → Multimedia streaming**.

KEYWORDS

360° video live streaming, 5G, delay measurements

ACM Reference Format:
Haitao Liu[1,2], Xinyi Zhang[1], Xiang Zhou[3], Tianyun Zhang[1],, Yanbiao Li[1,2], Zhenyu Li[2,4], Gaogang Xie[1,2]. 2022. When 360-degree Video Meets 5G: A Measurement Study of Delay in Live Streaming. In *17th ACM Workshop on Mobility in the Evolving Internet Architecture (MobiArch'22), October 21, 2022, Sydney, NSW, Australia.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3556548.3559632

*The corresponding author is Gaogang Xie, e-mail: xie@cnic.cn.

MobiArch'22, October 21, 2022, Sydney, NSW, Australia

ACM ISBN 978-1-4503-9518-2/22/10…$15.00
https://doi.org/10.1145/3556548.3559632

1 INTRODUCTION

360° video records views from each direction simultaneously by using an omnidirectional camera or a group of cameras, which provides viewers with an immersive viewing and interactive experience [1]. Video quality is an important factor that characterizes the viewer's QoE, especially when the viewer wears a VR headset to watch videos at close range. Due to the wide field of view, 360° video requires higher resolution (e.g., 4K, 8K or even 16K [2]) and bitrate to ensure high video quality. The high resolution leads to huge volume of data that requires higher network bandwidth to deliver the 360° video [3–6]. With the rapid development of live streaming technology, 360° video live streaming has been applied to many scenarios. In addition to high bandwidth, the requirement of low latency becomes more stringent under some interactive scenarios (e.g. interactive CloudVR [7]). For example, when the viewer interacts with the video source (e.g., switching viewpoint or changing position), a high motion-to-photon delay will cause viewer to feel dizzy [8] or disappointed. Therefore, the end-to-end delay plays a more important role in characterizing the viewer's quality of experience (QoE) especially when streaming a live 360° video.

Thanks to the high-bandwidth and low-latency nature, 5G is expected to pave the last mile when delivering 360° video live stream. To better understand the live streaming performance under 5G access, extensive delay measurement works should be conducted. However, both the camera and client access server via WiFi [9, 10], LTE [11–14], Ethernet [10] in most existing measurement works. Although Midoglu et al. [15] and Rigazzi et al. [16] proposed an evaluation framework or platform under 5G access, they didn't provide any measurement results about the live delay. Xu et al. [17] measured the frame delay under Non-Stand Alone (NSA) 5G access, which is the deployment mode in conjunction with 4G/LTE during the transition period and will not be widely used in the future. Therefore, the delay performance measurement of 360° video live stream under 5G access is not comprehensive enough yet.

In this paper, to further characterize the delay performance when delivering live 360° video under 5G access, we first build a 360° video live streaming system based on a commercial camera and a series of open-source projects, in which end devices access the server via our 5G network. Then, we measure the overall end-to-end live delay and find that 5G access indeed reduces the delay compared with WiFi or Ethernet access. To further explore what leads to the reduction, we measure the delay consumption of each stage in a breakdown way. By inserting timestamps at the entrance and exit of each stage, we analyze the server/client log and network trace and obtain the delay of each stage. According to the measurement results, we find that although the 5G upstream delay is the highest among that of WiFi and Ethernet, its lowest downstream delay still makes up the upstream delay and consequently contributes to the reduction of end-to-end live delay. In addition, we also find that the major delay bottlenecks are caused by the computation stage of the client, server and camera rather than the communication between them. In other words, the decoding/encoding process inside the client takes the most time, followed by the computation time of the server and camera.

The remaining parts of this paper are organized as follows. In Section 2, we overview the existing works. Our experimental settings are introduced in Section 3. We describe our measurement methodology in Section 4, and show the results in Section 5. Future directions are discussed in Section 6. Finally, we conclude this paper in Section 7.

2 RELATED WORK

There are several works focusing on measuring the 360° video live streaming system. We classify them into two categories, i.e., non-5G access and 5G access.

2.1 Non-5G Access

Yi et al. [10] measured the one-way delay and start-up delay of 360° video live streaming on YouTube platform. The camera pushed the stream to the server via WiFi access, and the client pulled from the server via Ethernet. Besides, Yi et al. [9] further measured each component of end-to-end live delay. In their Zeus prototype system, the cameras and client access the server via an University WiFi, and they found the major delay bottleneck is caused by the process when the server handles a connection request. Xie et al. [13] designed an adaptive compression scheme and measured video quality, frame delay and freezing ratio over LTE cellular networks. Lo et al. [14] measured video quality and transfer time under different bitrate and number of tiles over campus 4G network. Liu et al. [18] compared Broadcaster-to-Viewer delay of YouTube and Facebook platforms by employing crowd-source volunteers, but they didn't specify viewers' access types.

2.2 5G Access

All the above works have evaluated the streaming performance under WiFi, Ethernet, LTE or other access. To evaluate the streaming performance under 5G access, Midoglu et al. [15] proposed an evaluation framework for real-time adaptive 360° video streaming over 5G networks, but they didn't provide any measurement results. Similarly, Rigazzi et al. [16] built an edge and fog computing platform for the effective deployment of 360° video applications. They measured the GPU load, power consumption and memory usage on the cloud data center, and compare fronthaul/backhaul link data rate over different encoding schemes. Xu et al. [17] measured the frame delay and throughput of UHD Panoramic Video Telephony under 5G access. Compared with 4G, they found that 5G did reduce the frame delay to some extent, but they didn't uncover why this happened. Besides, their 5G network only supported NSA (Non-Stand Alone) mode, i.e., their 5G base stations are connected to 4G core network, which is not the current and future mainstream deployment mode for 5G.

3 EXPERIMENT SETUP

In this section, we describe the workflow, hardware/software design of our 360° video live streaming system.

3.1 Workflow

Our testbed and its workflow are shown in Figure 1 and Figure 2, respectively. The 5G access network consists of base station (5G gNB) and core network (5GC). Compared with Xu et al. [17], our 5G access network supports Stand-Alone (SA) mode. The camera and client access 5G network via two seperate RST CPEs (Customer Premises Equipments)[1]. CPE is a kind of 5G terminal. We adopt Insta360 Pro 2[2] camera as a video source. After capturing, stitching and encoding frames, camera streams the video to an RTMP (Real Time Messaging Protocol) server via our 5G network. Then, the Orientation server pulls the stream from RTMP server, when transcoding finished, it merges the tiles selected according to viewer's FoV (Field-of-View) feedback. The client receives the frames via 5G access and then renders them to display.

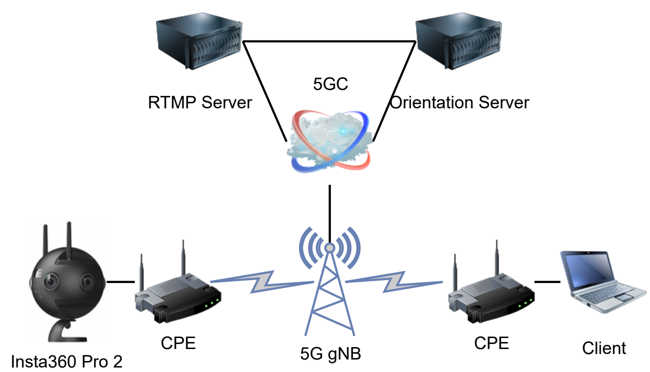

Figure 1: Overview of Streaming System under 5G Access

3.2 Hardware Configuration

In our 360° video live streaming system, 5GC, RTMP and Orientation server are deployed in virtual machines (VMs) hosted by the same physical server. Our client runs on a laptop. The CPU cores and memory allocated for each machine are listed in Table 1.

3.3 Software Configuration

Apart from hardware, our testbed is totally built on open-sourced software. We build our 5GC based on free5gc[3], which is the most popular open-source 5G core network solution. The RTMP server is built with the Nginx RTMP[4] module that supports pushing/pulling

[1] http://www.cd-rst.com/
[2] https://www.insta360.com/cn/product/insta360-pro2/
[3] https://github.com/free5gc/free5gc
[4] https://github.com/arut/nginx-rtmp-module

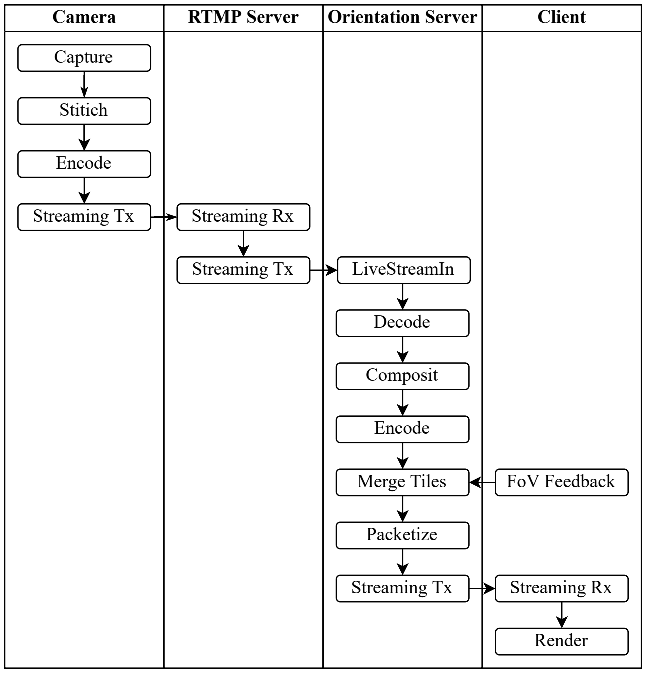

Figure 2: Live Streaming Workflow

Table 1: Hardware configuration

Component	CPU Info	Memory (GB)
5GC	8×AMD EPYC@2.2GHz	128
RTMP Server	4×AMD EPYC@2.2GHz	4
Orientation Server	8×AMD EPYC@2.2GHz	16
Client	4×Intel i3-3227U@1.9GHz	8

stream using RTMP. The Orientation server and client are built based on Immersive-Video-Sample[5] from Intel OpenVisualCloud project. The client communicates with Orientation server using WebRTC protocol.

4 MEASUREMENT METHODOLOGY

In this section, we first introduce how we measure the end-to-end live delay, then break it into several parts.

4.1 End-to-end Live Delay Measurement

The end-to-end live delay characterizes the viewer's perceived delay. Therefore, from the perspective of the viewer, we adopt the method used by Yu et al. [19]. Let the camera shoot a running stopwatch, then the client player will display the running stopwatch. We put these two stopwatches on the same screen and periodically capture the whole screen. Finally, we extract the timestamp value of stopwatches using tesseract-ocr[6] tool and calculate the difference between them. In this way, we obtain the end-to-end live delay.

[5]https://github.com/OpenVisualCloud/Immersive-Video-Sample
[6]https://github.com/tesseract-ocr/tessdoc

Table 2: Boundary function of each stage

Stage	Tx/Rx	Function name
RTMP Server	Rx	ngx_rtmp_receive_message
	Tx	ngx_rtmp_send_message
Ori. Server	Rx	LiveStreamIn::receiveLoop
	Tx	VideoFramePacketizer::onFrame
Client	Rx	WebRTCMediaSource::OnVideoPacket
	-	RenderManager::Render

4.2 Breakdown Measurement

4.2.1 Measure the In-Camera Delay. Following the live streaming pipeline in Figure 2, we first measure the In-Camera delay inside Insta360 Pro 2. It is worthwhile mentioning that Insta360 Pro 2 is a commercial camera, we cannot directly obtain the internal computation delay because inserting timestamp printing codes is not allowed. To tackle this problem, we treat all computation inside the camera as a whole, and use the same measurement method as described in subsection 4.1. First, the camera is directly connected to a control host, and their communication delay can be ignored. We run the official control software[7] of Insta360 Pro 2, then a preview video is displayed on the screen of control host. By capturing two stopwatches and calculating the difference, we can approximately obtain the In-Camera delay.

4.2.2 Measure the Cam2R Delay. We denote the communication delay between the camera and RTMP server as Cam2R delay. Since Insta360 Pro 2 is closed-source, we have no way to access the frame sending time. To evaluate the Cam2R delay, we instead measure the packet Round-Trip Time (RTT). Specifically, we insert a gateway host with two network interfaces between the camera and CPE. The gateway device is an Intel NUC [8] with 8 Intel i7-1165G7@2.8G cores and 16GB memory. We configure the IPv4 forward and NAT of gateway, and let the camera traffic enter gateway via one interface then leave it from another interface. Meanwhile, we capture the network trace on the gateway host and extract the ACK RTTs of packets.

4.2.3 Measure the Delay from RTMP Server to Client. From RTMP server on, the rest streaming pipeline is all open-sourced. We sort out the code of each stage, and find out the boundary function between the two stages. The boundary functions we have found are shown in Table 2. By printing timestamps from the boundary functions, we can record the ticks when a frame enters and exits each stage into log files, thereby calculating the computation delay in each stage and the communication delay between adjacent stages. In each experiment, the client renders about 2,000 frames when the measurement ends.

It is worth mentioning that the Orientation server does not process video frames strictly in order, and the composite step merges several video frames before delivering to next stage. This makes it difficult for us to establish correspondences of frames at different stages. To accurately measure the delay overhead, when a video

[7]https://www.insta360.com/cn/download/insta360-pro2
[8]https://www.intel.com/content/www/us/en/products/details/nuc.html

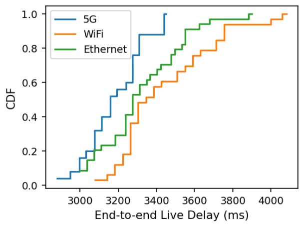

Figure 3: End-to-end Delay

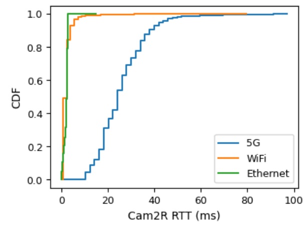

Figure 4: Cam2R Packet RTT

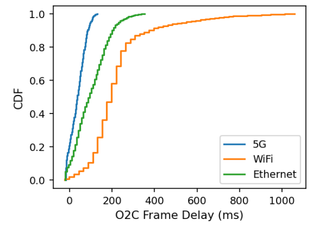

Figure 5: O2C Frame Delay

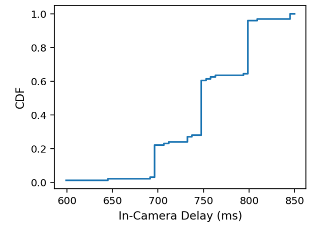

Figure 6: In-Camera Delay

frame enters the Orientation server, we attach a global ID to it to track the entire process of this frame in Orientation server.

5 MEASUREMENT RESULTS

In this section, we first introduce the end-to-end live delay, then we show the delay breakdown of each stage under three different access (i.e., 5G, WiFi and Ethernet).

5.1 End-to-end Live Delay

The CDF (Cumulative Distribution Function) of end-to-end live delay under three access networks is shown in Figure 3. We can find that 5G access achieves the lowest live delay compared with other two access, and the live delay under WiFi is the highest.

5.2 Delay Breakdown

5.2.1 Access Network Delay. As shown in Figure 1, different network access only affects the Cam2R RTT and O2C (Orientation server-to-Client) communication delay. Therefore, to uncover why end-to-end live delay achieves the lowest under 5G access, we compare the impact of 5G, WiFi and Ethernet access on the above two indicators. Figure 4 shows the Cam2R RTT delay under three access networks. Surprisingly, we see that most of Cam2R delay under 5G access are close to 40ms while those under other two access are close to 0ms. This indicates that 5G access behaves the worst upstream delay, respectively increasing 784.8% and 1259.8% compared with that of WiFi and Ethernet.

Now, we turn to the O2C delay shown in Figure 5. We see that 5G access outperforms than WiFi and Ethernet. Therefore, 5G access makes up for the upstream delay with ultra-low downstream delay, thereby reducing the end-to-end live delay.

5.2.2 Application Processing Delay. First, we take a look at In-Camera delay. As shown in Figure 6, the camera takes 700~800ms to capture, stitch and stream a video frame. Next, we compare the frame-level time consumption of each stage under 5G access in Figure 7. Besides, Table 3 records the total number of frames each stage delivers when measurement ends.

In Figure 7, we see that the client and Orientation server computation delay occupies the vast majority, and Orientation computation delay continually fluctuates. From Table 3, the client receives 2,029 frames but only renders 1,998 frames. In other words, the client takes time of 31 frames to decode and render a frame, this is corresponding to 1s computation delay since the FPS (Frame Per Second) of the video source is set to 30. We also find that the Orientation

Table 3: Frame Count of Each Stage

Stage	Rx/Tx	Frame Count
RTMP	Rx	2623
RTMP	Tx	2622
LiveStreamIn	Rx	2621
Decode	-	2520
Composit	-	2067
Encode	-	2034
Merge Tile	-	2034
Packetize	Tx	2034
Client	Rx	2029
Client render	-	1998

computation delay is relatively high. This is because the decoding, compositing and encoding processes cost considerable time. As shown in Table 3, when measurement ends, Orientation server receives 2,621 frames (`LiveStreamIn` Rx), but only delivers 2,034 frames (`Packetize` Tx) to client. There are 587 frames still left in the pipeline of Orientation process, especially the `Composit` and `Encode` steps.

In addition, at the very beginning of streaming, the R2O communication delay (i.e., the communication delay between RTMP and the Orientation server) decreases sharply. This is probably because many frames leave RTMP server in order when R2O connection is established. When the connection is established, this set of frames are burstly delivered to Orientation server almost at the same time.

At last, the RTMP computation delay is extremely low, which takes only 1 frame. In other words, it does not constitute the delay bottlenecks of the pipeline.

Furthermore, we show the frame-level live delay breakdown under WiFi and Ethernet access in Figure 8 and Figure 9, respectively. The similar phenomenon can be also seen. The only difference is that the client computation delay under WiFi access fluctuates significantly. This may be because the O2C delay of WiFi access is too large, causing the client to spend a certain amount of time waiting for the adjacent video frames to arrive and prepare for rendering. In the future, we will further explore the impact of O2C delay on client computation delay.

6 FUTURE DIRECTIONS

From our measurement analysis, we have proved that 5G access can reduce end-to-end live delay due to its lowest downstream delay.

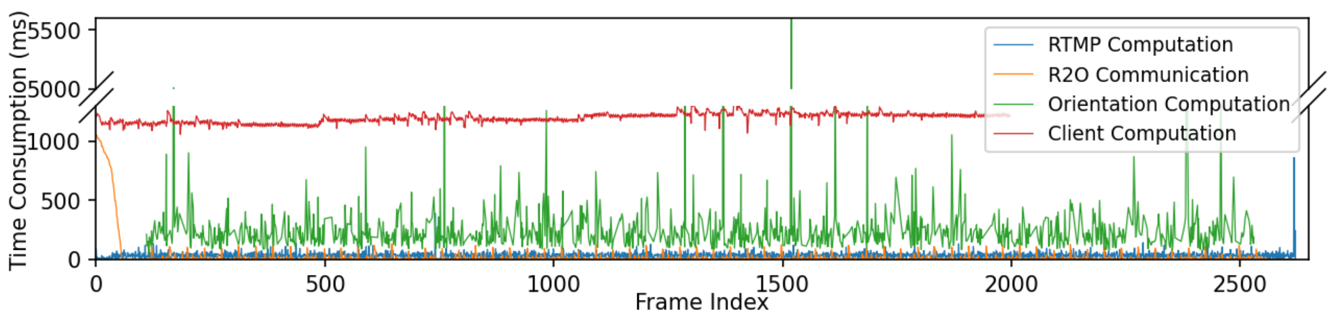

Figure 7: Frame-Level Live Delay Breakdown under 5G Access

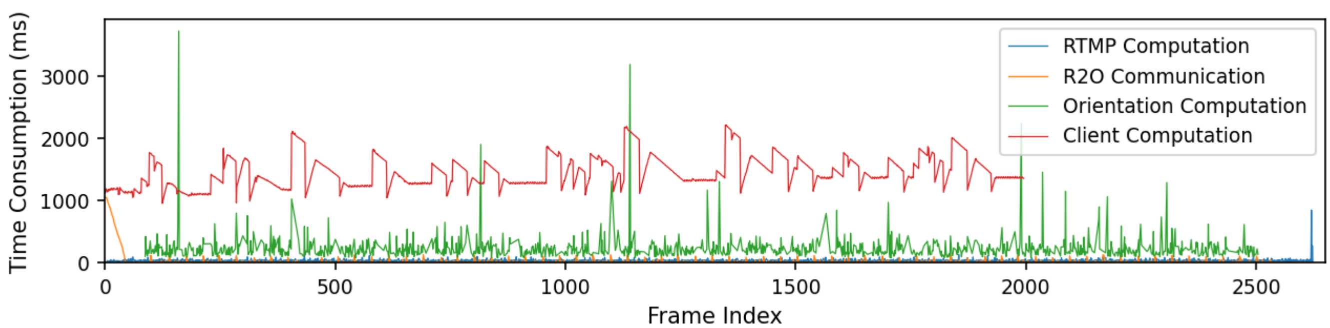

Figure 8: Frame-Level Live Delay Breakdown under WiFi Access

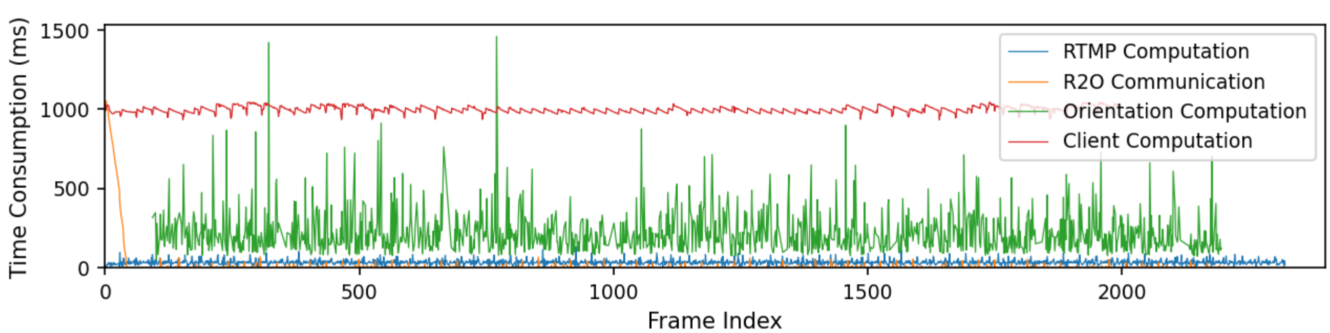

Figure 9: Frame-Level Live Delay Breakdown under Ethernet Access

In addition, we also located the major bottlenecks of 360° video live streaming, i.e., the computation delay of client, Orientation server and camera. Based on these observations, we will pay attention to the following directions in the future:

- Dive into the upstream delay under 5G access. We have seen that the upstream delay under 5G access is the highest among that of WiFi and Ethernet. Therefore, we plan to dive into the upstream delay and identify where the bottleneck locates.
- Improve hardware configuration. We have already identified three major delay bottlenecks lying in the application computation. To reduce the computation delay, we plan to equip Orientation server and client with GPUs to speed up their encoding/decoding process. Unfortunately, we can't improve the hardware capability of the Insta360 Pro 2, but we can advise manufacturers to do so.
- Expand to multi-viewer scenario. In this work, there is only one viewer watching the 360° video live. If multi-viewers simultaneously watch the live video, whether our 5G network becoming congested remains to be explored.

7 CONCLUSION

In this paper, we introduce our 360° video live streaming system with 5G access and measure the end-to-end live delay in both overall and breakdown way. We find that 5G access indeed reduces the end-to-end delay compared with WiFi and Ethernet access. In addition, the major delay bottlenecks of the streaming pipeline are caused by the computation of the client, Orientation server and camera. Based on our measurement analysis, we propose the directions for future research.

ACKNOWLEDGEMENT

This work is supported by the National Key R&D program of China (Grant No. 2019YFB1802800) and the National Science Fund for Distinguished Young Scholars (Grant No. 61725206).

REFERENCES

[1] Abid Yaqoob, Ting Bi, and Gabriel-Miro Muntean. A survey on adaptive 360° video streaming: Solutions, challenges and opportunities. *IEEE Communications Surveys Tutorials*, 22(4):2801–2838, 2020.

[2] Wenxiao Zhang, Feng Qian, Bo Han, and Pan Hui. Deepvista: 16k panoramic cinema on your mobile device. In *WWW*, 2021.

[3] Xianda Chen, Tianxiang Tan, and Guohong Cao. Popularity-aware 360-degree video streaming. In *IEEE INFOCOM 2021*, pages 1–10, 2021.

[4] Lovish Chopra, Sarthak Chakraborty, Abhijit Mondal, and Sandip Chakraborty. PARIMA: viewport adaptive 360-degree video streaming. In *WWW '21*, pages 2379–2391, 2021.

[5] Feng Qian, Bo Han, Qingyang Xiao, and Vijay Gopalakrishnan. Flare: Practical viewport-adaptive 360-degree video streaming for mobile devices. In *MobiCom*, pages 99–114, 2018.

[6] Ghufran Baig, Jian He, Mubashir Adnan Qureshi, Lili Qiu, Guohai Chen, Peng Chen, and Yinliang Hu. Jigsaw: Robust live 4k video streaming. In *MobiCom*, pages 1–16, 2019.

[7] Teemu Kämäräinen, Matti Siekkinen, Jukka Eerikäinen, and Antti Ylä-Jääski. Cloudvr: Cloud accelerated interactive mobile virtual reality. In *MM 2018*, pages 1181–1189. ACM, 2018.

[8] Kevin Boos, David Chu, and Eduardo Cuervo. Flashback: Immersive virtual reality on mobile devices via rendering memoization. In *MobiSys*, pages 291–304, 2016.

[9] Jun Yi, Md Reazul Islam, Shivang Aggarwal, Dimitrios Koutsonikolas, Y Charlie Hu, and Zhisheng Yan. An analysis of delay in live 360° video streaming systems. In *MM*, pages 982–990, 2020.

[10] Jun Yi, Shiqing Luo, and Zhisheng Yan. A measurement study of youtube 360° live video streaming. In *NOSSDAV 2019*, pages 49–54.

[11] Lei Zhang, Yanyan Suo, Ximing Wu, Feng Wang, Yuchi Chen, Laizhong Cui, Jiangchuan Liu, and Zhong Ming. Tbra: Tiling and bitrate adaptation for mobile 360-degree video streaming. In *MM*, 2021.

[12] Afshin Taghavi Nasrabadi, Anahita Mahzari, Joseph D Beshay, and Ravi Prakash. Adaptive 360-degree video streaming using scalable video coding. In *MM*, pages 1689–1697, 2017.

[13] Xiufeng Xie and Xinyu Zhang. Poi360: Panoramic mobile video telephony over lte cellular networks. In *CoNEXT*, pages 336–349, 2017.

[14] Wen-Chih Lo, Ching-Ling Fan, Shou-Cheng Yen, and Cheng-Hsin Hsu. Performance measurements of 360 video streaming to head-mounted displays over live 4g cellular networks. In *APNOMS*. IEEE, 2017.

[15] Cise Midoglu, Özgü Alay, and Carsten Griwodz. Evaluation framework for real-time adaptive 360-degree video streaming over 5g networks. In *Proceedings of the 2019 on Wireless of the Students, by the Students, and for the Students Workshop*, pages 6–8, 2019.

[16] Giovanni Rigazzi, Jani-Pekka Kainulainen, Charles Turyagyenda, Alain Mourad, and Jaehyun Ahn. An edge and fog computing platform for effective deployment of 360 video applications. In *WCNCW*, 2019.

[17] Dongzhu Xu, Anfu Zhou, Xinyu Zhang, Guixian Wang, Xi Liu, Congkai An, Yiming Shi, Liang Liu, and Huadong Ma. Understanding operational 5g: A first measurement study on its coverage, performance and energy consumption. In *SIGCOMM*, pages 479–494, 2020.

[18] Xing Liu, Bo Han, Feng Qian, and Matteo Varvello. Lime: understanding commercial 360 live video streaming services. In *MMSys*, 2019.

[19] Chenguang Yu, Yang Xu, Bo Liu, and Yong Liu. "can you see me now?" a measurement study of mobile video calls. In *INFOCOM*, pages 1456–1464. IEEE, 2014.