An Experimental Evaluation of 360-Degree ABR Video Streaming over mmWave Wireless Links

Sam Shippey[1], Suresh Srinivasan[1], Huu Phuoc Dang[2], Ehsan Aryafar[1], Jacob Chakareski[2]
[1]Portland State University, USA; [2]New Jersey Institute of Technology, USA

***Abstract*—Video streaming is a key technology for the envisioned Metaverse. Present wireless technologies have limited throughput, which limits the quality of video content (e.g., 360◦ video) that can be delivered to mobile clients. To address this challenge, wireless carriers have begun to deploy millimeter wave (mmWave) networks, which have high throughput potentially in the multi-gigabit per second range. However, mmWave wireless is highly sensitive to blockage and mobility that induce dramatic variations in link-level throughput. This can in turn penalize the video quality of experience delivered to clients. This adverse effect can potentially be mitigated by the use of adaptive bitrate (ABR) techniques that dynamically adjust the bitrate of the video content based on network conditions and device capabilities. However, existing studies of ABR video streaming performance over mmWave wireless links assume channels with low throughput, use simulations (with synthetic or measured traces) for performance evaluation, and/or conduct large scale experiments that mask client performance at the individual level. In this paper, we develop a testbed to experimentally assess 360° video ABR performance of three prominent methods over high capacity (high throughput) mmWave wireless links for a variety of channel conditions. We show that the fundamental tradeoff between stalls and visual quality becomes much sharper at higher throughput, and that existing ABR methods do not achieve substantial gains in one without losses in the other. For instance, we find that complex ABR strategies which outperform simpler strategies in terms of visual QoE also increase stall time from less than 0.5 seconds to nearly 4.**

I. Introduction

Wireless communication plays a vital role in enabling Metaverse and VR/AR applications and the 3GPP (cellular wireless standardization organization) has identified Metaverse and VR/AR applications as key emerging applications for 5G+/6G technologies [1]. Recent reports [2], [3] indicate that the necessary wireless data rates for Metaverse video range from 15 Mbps for a 4K 30fps H.264-encoded 360° video stream to 800 Mbps for an 8K H.266 encoded stream, and even up to several Gbps for higher resolutions and frame rates [4]. Additionally, MPEG suggests a minimum requirement of 12K spatial resolution and 100 frames per second (fps) temporal display rate for a 360° video panorama experienced by a VR client [5]. Even with state-of-the-art High Efficiency Video Coding (HEVC) compression, meeting these demands would still necessitate data rates in the range of several Gbps.

Existing wireless technologies such as LTE and sub-6 GHz 5G often struggle to achieve these data rates. For instance, LTE typically offers speeds around 10 Mbps, which fall significantly short of the data rates required for the Metaverse. Thus, majority of wireless operators have begun deploying high frequency mmWave networks, which can provide high capacity due to the large amount of available spectrum/bandwidth. While existing mmWave 5G deployment use smaller channels for communication with each client[1], next generation 5G+/6G wireless networks are expected to substantially increase the amount of bandwidth (and hence data rate), available to each client. However, mmWave communication faces challenges such as mobility management and susceptibility to blockages. The mobility issue is because mmWave systems use highly directional beams to combat the high path loss associated with higher frequencies. This makes beam alignment between a transmitter and receiver and beam tracking a challenge. The blockage issue is an inherent problem associated with the frequency band, e.g., human body alone can block the signal by more than 20 dB [7], [8]. As a result, high throughput mmWave systems are expected to face large fluctuations in data rate in a variety of environments, such as shopping malls, downtowns, and even gaming arcades.

Large fluctuations in throughput can reduce the Quality of Experience (QoE) of the client. Existing video providers/players leverage ABR mechanisms to provide a seamless viewing experience by adjusting the video quality in real-time to match the viewer's internet connection speed and device capabilities. ABR mechanisms have shown promising performance in existing LTE, sub-6 GHz 5G, and small bandwidth mmWave 5G systems, but a key question that remains to be answered is *how do existing ABR methods operate in high bandwidth mmWave systems, with both much higher peak throughput and much greater variation in throughput?* In this paper, we develop an open-source testbed to experimentally answer this question. Specifically, we make the following contributions:

- **System Development.** We develop a system, based on iStream Player [9] and commercially available mmWave equipment, to ingest and stream high resolution 360° videos, while reporting a variety of performance metrics. We have updated the player to include LoL$^+$ [10], a new learning-based ABR method, as well as several libraries to generate and process high bitrate video.
- **Performance Evaluation.** We conduct extensive evaluation of three prominent ABR streaming strategies in order to gain an understanding of how the unique challenges which occur in highly variable wireless environments im-

[1]For example, AT&T's 5G network uses channels of 100 MHz bandwidth for each client [6].

pact video streaming performance. We find that existing methods of increasing video quality can increase stalls by up to a factor of 10, while methods of decreasing stalls can lead to a decrease in selected video bitrate by 34%. The gap between different performance objectives widens with more fluctuations in throughput, e.g., when a client is mobile and frequently changes its channel condition from Line-of-Sight (LoS) to non-Line-of-Sight (nLoS).

The paper is organized as follows. We discuss the related work in Section II. Section III provides a brief background and motivation for our research problem. We introduce our experimental setup and implementation in Section IV. In Section V, we analyze the results of our extensive experiments. Finally, we present concluding remarks in Section VI.

II. Related Work

We present a brief overview of ABR mechanisms and their evaluation over wireless systems.

ABR streaming strategies. There have been many strategies developed over the years for selecting the bitrate at which a client will select a video quality. They generally fall into 4 groups: (i) Purely buffer-based strategies such as BBA [11] and BOLA [12]; (ii) Bandwidth-aware strategies such as Dynamic [13], FESTIVE [14], and PANDA [15]; (iii) Control-theoretic or online learning-based approaches which do not incorporate reinforcement learning, starting with the adoption of FastMPC [16] and later strategies such as LoL$^+$ [10] and L2A [17], which use Kohonen maps and online optimization respectively to keep up in scenarios requiring low latency; and (iv) Reinforcement learning (RL) based approaches, most notably Pensieve [18]. RL-based approaches have become popular in recent years, but can also require large amounts of either historical streaming performance (not just throughput) data or extensive simulation to train. We evaluate three ABR methods in this paper, including buffer-based, bandwidth-based, and learning-based methods.

Adaptive streaming over mmWave. Struye et. al. [19] consider streaming video for VR devices over mmWave but do not consider multiple methods of adaptive streaming nor performance under client mobility. Le et. al. [20] considers real time VR video streaming but does not cover adaptive streaming, instead focusing on offloading. mmWave links integrated in parallel dual connectivity streaming systems have been explored to enable Gbps data rates needed for lifelike VR immersion [21]–[24], but they do not consider integration with ABR strategies. Storck and Duarte-Figueiredo [25] conduct an evaluation similar to our proposed methodology but focus exclusively on outdoor vehicular traffic as opposed to our more controlled indoor one. Ramadan et. al. [26] study ABR streaming over mmWave 5G but use traces rather than a real world testbed for evaluation, as well as using much smaller, non-360° videos. Arunruangsirilert et. al. [27] consider 5G specific ABR streaming by retraining Pensieve [18] on 5G traces but they only conduct simulations with much smaller channel bandwidth. Finally, Yan et. al. developed Puffer [28] in order to conduct a large scale analysis of ABR streaming in the wild, but this is not specific to mmWave clients. In short, we have not yet found a comprehensive study that experimentally characterizes 360° video ABR streaming over large bandwidth (data rate) mmWave links across a variety of channel conditions. This paper aims to fill this gap.

III. Background and Motivation

A. Preliminaries

Quality of Experience. Video data is complex and varies across many different parameters. For instance, an encoder may select a high or low bitrate or resolution. Video streaming is even more complex, encompassing parameters such as segment length, ABR strategies, and throughput estimation. To standardize our measurements we leverage two commonly used QoE axes: Visual quality and stalls. Measuring stalls is straightforward, but there are many metrics for measuring video quality, including using the video's bitrate, perceptual metrics such as PSNR and SSIM [29], and fusional methods such as VMAF [30]. We use bitrate, VMAF, and PSNR as visual QoE metrics, and also include a composite QoE metric commonly used in evaluating ABR strategies and sometimes used as an optimization target in strategies which choose to frame adaptive streaming as an optimization problem. This composite QoE can be expressed as [16]:

$$QoE_C^K = \sum_{k=1}^{K} q(R_k) - \lambda \sum_{k=1}^{K-1} |q(R_{k+1}) - q(R_k)| - \mu \sum_{k=1}^{K} \left(\frac{d_k(R_k)}{C_k} - B_k \right) \quad (1)$$

Where R_k is the selected video bitrate at time k, $q(R_k)$ is a quality function (usually linear or log) that captures the impact of video compression [31], $d_k(R_k)/C_k$ is download time at rate C_k and B_k is seconds of video remaining in the buffer. The first term represents video quality, the second term a penalty for frequent quality switches, and the third term imposes a penalty for rebuffering (how many seconds download time exceeds the amount of video buffered). How much to penalize is decided by the parameters λ and μ, respectively. We used two $q(R_k)$ functions, a linear and log function of bitrate, and set λ and μ to 3 and 8 (4 in the logarithmic case) respectively, in our performance evaluation results of Section V.

Adaptive bitrate video streaming. Network fluctuations are common in networks. Everything from congestion to handoff can influence end-to-end throughput. As such, video streaming services often attempt to adapt to changing network conditions by adjusting the bitrate at which they request content. For instance, when a connection is good, the client can request 1080p or 4K content, but when a connection is bad the client will automatically switch to a lower quality until the network recovers. This paradigm is known as adaptive streaming and has seen rapid growth in the last decade alongside the increase in demand for video content,

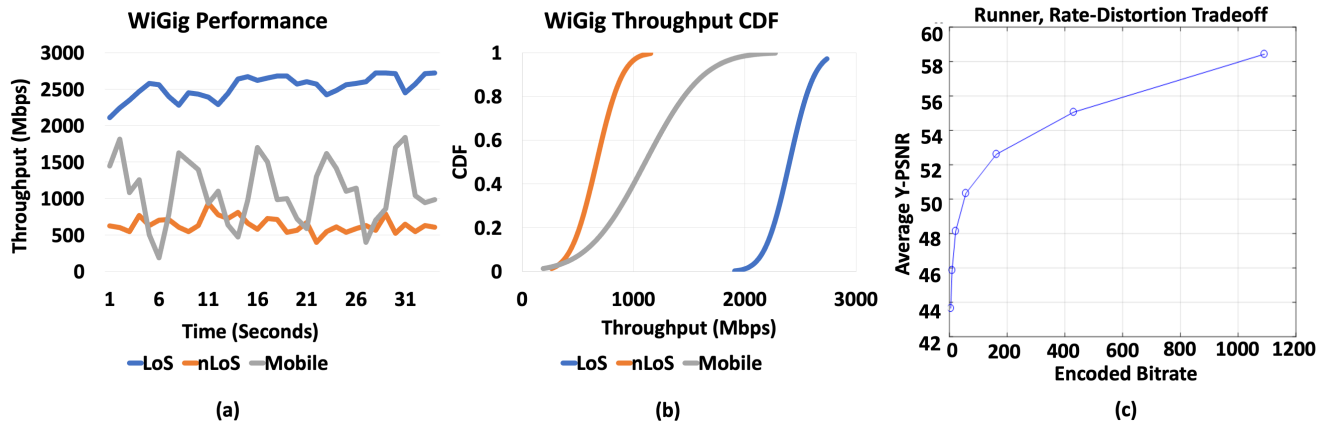

Fig. 1. **(a): WiGig performance as a function of time, measured by saturating the link with iPerf. The three channel conditions refer to stationary Line of Sight (LoS), stationary non-Line of Sight (nLoS), and a mobility pattern created by moving in a square, alternating LoS and nLoS. Note the large fluctuations in the mobility case as the transmitter passes in and out of the AP's LoS. (b): CDF curve of the same channel conditions. (c): Rate distortion characteristics for Runner, one of the videos we use in our evaluations. Variations in bitrate can cause large fluctuations in PSNR.**

especially in scenarios where human users would be unable to adjust the bitrate before rebuffering occurs, such as in low latency livestreaming. Different ABR strategies have different assumptions about network conditions and QoE targets, and consider different information. For instance, a simple ABR strategy may consider only current estimated network throughput. Further, many implementations of ABR streaming exist, such as DASH.js [32] and iStream Player [9]. While many implementations focus on adaptive streaming over HTTP, the adaptive streaming paradigm is separate from frameworks such as DASH.js and even standards such as MPEG-DASH or HLS, adaptive bitrate strategies can be used across other protocols, such as WebRTC [33] or Media over QUIC [34].

B. Motivation

We conducted preliminary experiments to demonstrate large fluctuations in throughput associated with high throughput mmWave systems and the potential impact on perceived video visual quality. Our setup consists of a Netgear Nighthawk X10 router with a WiGig wireless radio connected to a Dell server over 10G Ethernet. WiGig or 802.11 ad is an unlicensed communication technology developed by IEEE. Our WiGig radios use communication channels of 2 GHz bandwdith centred around 60 GHz center frequency. Our client device is an Acer TravelMate laptop equipped with a similar WiGig wireless card. Positioned atop a TurtleBot robot, this laptop can be programmed for either stationary or mobile operation with customizable speed and mobility patterns. These devices are deployed within an indoor office environment.

We examine three distinct scenarios: (i) Line-of-Sight (LoS): In this configuration, the client remains stationary approximately 2 meters away from the WiGig base station (BS). The client has a direct unblocked path to the WiGig BS and remains fixed throughout the experiment. (ii) Non-Line-of-Sight (nLoS): In this arrangement, the client remains stationary at a location similar to the LoS experiment, but a human blocker stands approximately 1 meter in front of the WiGig BS throughout the experiment, blocking the direct path between the client and the BS. (iii) Mobility: In this setup, the client device (TravelNate laptop positioned on top of the Turtlebot robot) traverses a rectangular path measuring 6 ft by 2 ft. There is no human obstruction between the client and the BS. Initially, the client faces the BS directly in a LoS channel condition. However, as the robot turns 90°, followed by two additional 90° turns, the client's channel condition transitions to nLoS because the laptop's body obstructs the LoS path. Consequently, in this setup, the client frequently alternates between LoS and nLoS channel conditions, capturing both the impact of mobility and blockages.

Iperf throughput and PSNR evaluation. We assess TCP performance using iperf on both the client (Travelmate) and server, recording throughput every second over a span of 10 minutes across the three aforementioned channel conditions. Fig. 1(a) and (b) display the throughput variations observed under these different channel conditions. In Line-of-Sight (LoS), the average throughput reaches approximately 2.4 Gbps with small fluctuations around this average value. However, we observe large fluctuations in the mobility scenario, which includes both the impact of blockage and mobility. The two challenges would require beam tracking as well as frequent insertions of sector sweeps to find and align beams. As a result, the immediate throughput between the client and BS can be as low as 0.25 Gbps and as high as 1.8 Gbps. We also observe that in nLoS scenario, the average throughput is approximately 0.6 Gbps. Note that even in stationary LoS and nLoS scenarios with no blockages, there is still noticeable fluctuations in throughput. We found through careful examination of traces that even in these scenarios the selected beams can frequently change[2], resulting in changes in measured throughput.

Subsequently, we conduct MATLAB simulations using a

[2]This is because the IEEE 802.11 ad standard by default performs the beam search process every beacon interval, which can change the selected beams for both the BS and client.

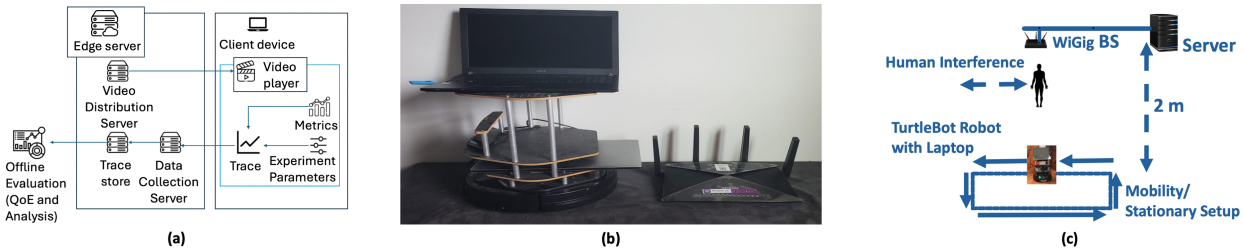

Fig. 2. (a): Different components of the software implementation. The video is sent from the video distribution server on the edge server to the client device's video player. The input traces are then subjected to tests with different metrics and experimental parameters. Finally, the results are saved on a remote server prior to an offline QoE evaluation and analysis. (b): Our physical testbed, consisting of a laptop with a WiGig card, a WiGig base station, and a TurtleBot robot. (c): In nLoS experiments, a human blocker stands in front of the BS (at one meter away), blocking the direct path between the client and the BS. In mobility experiments, the client is placed on a Turtlebot robot with controlled mobility pattern and speed. The client traverses a 6 ft by 2 ft rectangular pattern. The client WiGig radio is positioned in the front of the laptop, thus as the laptop turns, its body blocks the LoS path to the BS.

known video (Runner). Fig. 1(c) illustrates the Quality of Experience (QoE) in terms of Y-PSNR relative to bitrate. The QoE of the runner video notably improves as bitrates are increased, further highlighting the necessity for reliable high throughput over mmWave to achieve superior QoE.

IV. Implementation and Measurement Setup

We conduct our experiments using a set of extremely high bitrate videos, streaming using MPEG-DASH from an edge server to a client over a single WiGig BS. To stream the videos, we use a modified version of iStream Player [9], which includes an implementation of LoL$^+$ [10] and a video ingestion and preparation step. Fig. 2(a) shows the different software components of our setup. We plan to publicly release all of our software and gathered data upon paper acceptance.

Network Deployment. Our hardware is composed of Nighthawk X10 router, Dell server, TravelMate laptop, and Turtlebot robot, a discussed in Section III. These devices are shown in Fig. 2(b) and are deployed in an indoor office environment as depicted in Fig. 2(c). We conduct experiments under the three channel conditions discussed in Section III. We use TCP/IP as the underlying transport protocol.

Video Player. We initially planned to evaluate performance using a web-based player such as DASH.js. However, when we decided to use extremely high bitrate videos, we discovered that the Media Source Extensions buffer is insufficient to display such content even when using very small segments. iStream Player is easy to modify and most importantly capable of headless operation, allowing us to bypass the limitations imposed by small source buffers or slow decoder implementations faced in browsers.

iStream Player does not include a video ingestion system to prepare videos for streaming, so we added one to make preparing videos for streaming and evaluation easier. This feature makes it easy to pass videos of several kinds (H.264 or H.265, VP9, or even source) into iStream Player and automatically handle bitrate ladder creation, visual quality of experience evaluation, and dashification process. We implement a simple static bitrate ladder creation process optimized for evaluating the kinds of short video streaming we evaluate in this paper, but since we have taken care to respect iStream Player's initial design goal of extensibility and easy modification, dynamic or optimizing bitrate ladder generation could be added as needed by others as well. We have also produced an implementation of LoL$^+$ within iStream Player.

Video dataset. We use 8K 360° videos encoded at bitrates between 150 Mbps and 1.5 Gbps. We source the videos from the publicly available SJTU 8K 360° video dataset [35]. We encode each video in the dataset (labeled Runner, Academic, and Kaimenxueye) as standard H.264 videos at a variety of bitrates. In each case we keep the GOP[3] size fixed at 30 frames by forcibly inserting keyframes every 30 frames. We use a strictly enforced constant bitrate when encoding the videos to ensure smooth performance over the result of the streaming session. We set the segment size to 1 second when preparing the videos for streaming. We use 8 bitrate targets: 150, 250, 450, 550, 650, 900, 1200, and 1500 Mbps. Irrespective of the selected bitrate, the duration of each video is 36 seconds.

ABR video streaming. In this evaluation, we consider three well-known ABR strategies which are representative of three broad approaches to bitrate adaptation: Average-bandwidth, buffer-based [11] and LoL$^+$ [36]. These three strategies are simple to explain and reason about, while still providing insight on the performance of other adaptive bitrate strategies.

Bandwidth. Bandwidth simply estimates the available bandwidth for the next transmission by taking an exponentially weighted moving average over bandwidth history. To obtain bandwidth history, the ABR strategy waits until the previous download has finished and uses its size and download time to estimate available bandwidth. The algorithm then requests the appropriate quality level for the next video segment based on this value. Bandwidth estimation is a challenging task as it is sensitive to many different processes in the network and parameters of the video streaming system, such as how long a segment is. Shorter segments mean more immediate updates, but may cause the system to over or underestimate bandwidth if a change in network conditions is short. To make matters worse, it is rare that a video streaming system can actually saturate a link, meaning the ABR strategy may perform sub-

[3]In video coding, a group of pictures, or GOP structure, specifies the order in which intra- and inter-frames are arranged. The GOP is a collection of successive pictures within a coded video stream.

optimally, especially in the high fluctuation range of high-frequency wireless. As such, we use this strategy to evaluate the impact of the highly unstable link on ABR performance.

Buffer-based Algorithm (Buffer). Buffer [11] fills the playback buffer with low quality segments until a reservoir duration is reached, then ramps up quality until a cushion is established. After the cushion is established, the player requests the maximum rate. This algorithm completely ignores rate estimation and uses exclusively the buffer level to figure out the next segment's quality. It focuses primarily on reducing stall time and startup delay.

Low on Latency+ (LoL$^+$). LoL$^+$ [10] is an extension of the Low on Latency authors' previous approach [36], which builds on a more general strategy for model predictive control called FastMPC [16]. The basic idea is to choose the best choice from a permutation of all possible ABR choices on an optimization horizon. LoL$^+$ makes the LoL algorithm more configurable to allow it to accommodate for some difficult edge cases. We implement LoL$^+$ within iStream Player in order to test it using the ultra-high bitrate videos we prepared.

V. Results and Discussion

We present the results of our experiments in Tables I and II. Table I gives results averaged over all videos in the dataset, while Table II gives averages over 20 trials per combination of ABR strategy and channel condition for each specific video. We present results in terms of three visual quality of experience metrics (VMAF, PSNR, and bitrate) in order to remove ambiguities due to measurement. We also present results showing the average number of stalls and the average durations of those stalls (shown as Time of Stalls in our tables).

TABLE I
Averaged metrics over all videos. The best results for a given channel condition are bolded.

Mobility		Stationary		Mobile
Line of Sight		LoS	nLoS	
Bandwidth	VMAF	96.94	96.77	96.5
	PSNR	55.23	54.26	53.06
	Bitrate	1017	818.18	636.27
	# of stalls	1.00	1.00	1.02
	Time of Stalls	0.31	0.32	0.41
	Log QoE	75.0	64.4	44.9
	Linear QoE	334.9	247.0	158.1
Buffer	VMAF	96.98	95.59	96.71
	PSNR	55.45	54.06	53.85
	Bitrate	1071.00	810.49	749.98
	# of stalls	1.00	1.00	1.00
	Time of Stalls	**0.25**	**0.28**	**0.37**
	Log QoE	74.1	**66.9**	**55.0**
	Linear QoE	337.1	268.7	183.5
LoL+	VMAF	**97.25**	**96.96**	**96.96**
	PSNR	**56.51**	**55.07**	**55.10**
	Bitrate	**1270.00**	**982.9**	**974.8**
	# of stalls	1.00	1.15	2.58
	Time of Stalls	1.55	1.94	3.75
	Log QoE	**80.5**	64.6	51.4
	Linear QoE	**406.9**	**272.1**	**203.6**

Additionally, we break our results down in terms of 3 commonly used axes in video streaming: average visual quality of experience, stall time, and quality switches. These are commonly combined in the composite QoE metric presented in Eq. (1). We present two composite QoE values in our tables: log and linear based. In both of these metrics, we use bitrate as the visual quality of experience and combine it with other parameters according to Eq. (1).

From Table I, we observe that LoL+ provides a higher average visual QoE in all three channel conditions (stationary LoS, stationary nLoS, and mobile) across VMAF, PSNR, and bitrate, which results in a higher composite linear QoE as well. Buffer has a better performance in terms of stall time (duration) across all three channel conditions, which results in a higher composite logarithmic QoE in nLoS and mobility scenarios. This is because video stalls are much more common in nLoS and mobility channel conditions. However, we observe that in LoS channel conditions, LoL+ has a higher composite logarithmic QoE (with bandwidth having the second highest performance). This is because of the much higher (about 20%) improvement in visual quality in terms of bitrate of LoL+ compared to Buffer. Table II shows the values for the same metrics for each of the three videos studied in this paper. We observe some variations in results based on the specific video. For example, we observe that for the Academic video the Buffer method provides the highest visual QoE (VMAF, PSNR, and bitrate) as well as a higher linear and logarithmic composite QoE.

Findings: QoE is highly dependent on the channel condition as well as the specific video. A context-aware solution that can infer channel condition (LoS, nLoS, and mobile) has the potential to significantly boost QoE in mmWave systems by adapting the selected ABR strategy based on the channel condition and the specific video.

Visual Quality of Experience. We next present temporal graphs by plotting visual quality of experience over time in terms of average achieved video bitrate (the rate selected by the ABR strategy), PSNR, and VMAF in Fig. 3. These results are divided based on the channel condition.

All three metrics are in agreement although the differences are easiest to observe using achieved video bitrate and PSNR since VMAF scores become very similar when using very high bitrate encoding. Bandwidth and LoL$^+$ are stable after their startup phases in the Line-of-Sight case, while Buffer periodically plays out the buffer and drops the quality by about 1 dB to compensate. In contrast, the periodic drop and recovery in the mobile case is visible for all 3 strategies and is most severe, about 2 dB in video PSNR or 150 Mbps difference in video bitrate, using LoL$^+$. Bandwidth and Buffer both maintain more consistent bitrate selections, differing generally less than 1 dB PSNR, despite $>$ 700 Mbps fluctuations at the transport layer throughput as shown in Fig 1. In the non-Line-of-Sight case, the visual quality of experience is noticeably lower but stable as expected. Despite larger fluctuations, LoL$^+$ performs the best consistently on visual quality, followed by Buffer, then Bandwidth. LoL$^+$ performs the best on visual quality of experience since it directly optimizes for achieved video bitrate, which serves as a good proxy for PSNR and VMAF.

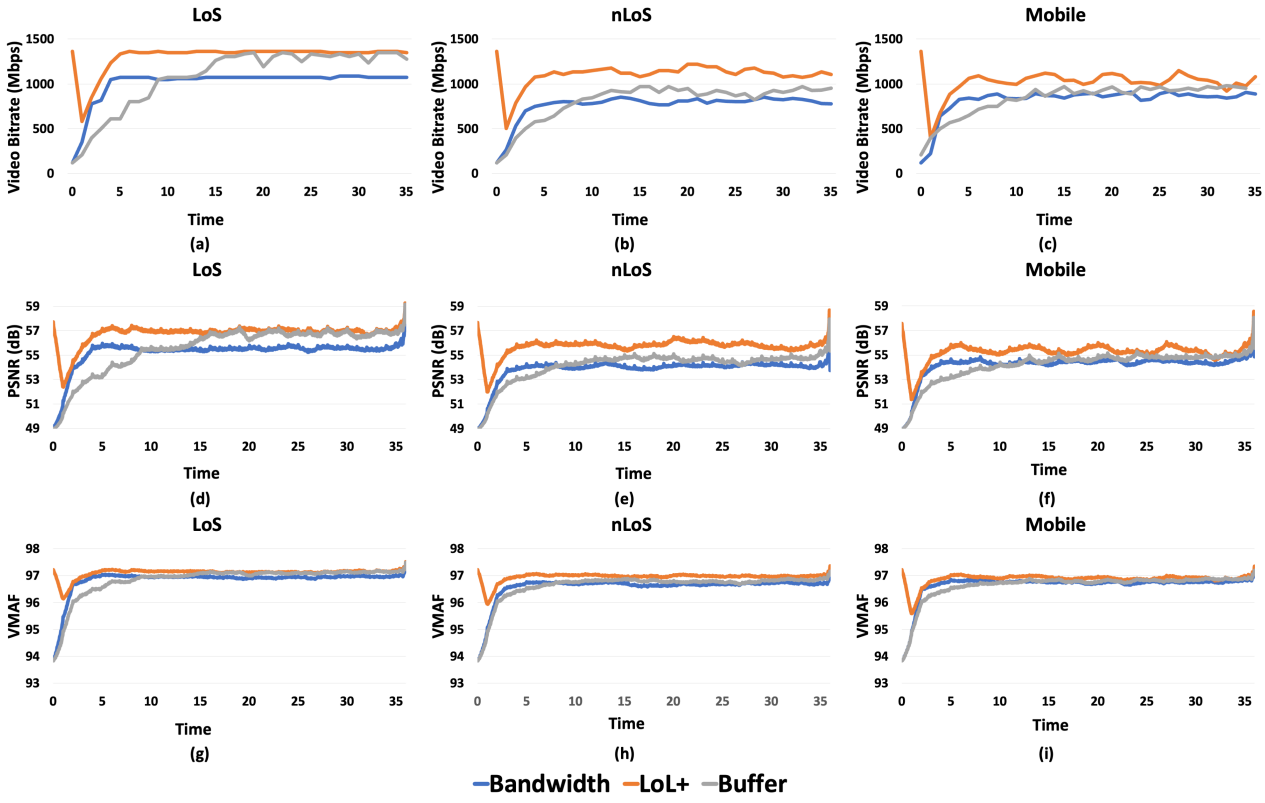

Fig. 3. **(a-f): Average results across all videos in terms of visual quality of experience over time (in seconds). Recall that the duration of each video is 36 seconds. Figures a-c show results in terms of achieved video bitrate, d-f in terms of PSNR, and g-i in terms of VMAF. LoL$^+$ has an optimistic startup phase, whereas Buffer and Bandwidth underestimate the video quality that can be transmitted. The mobile scenario has larger fluctuations.**

The difference between Buffer and Bandwidth's performance mostly comes down to Buffer being able to take advantage of buffering video and download higher quality segments.

Findings: LoL$^+$, by virtue of directly optimizing for video bitrate, maintains high quality even in the presence of large bandwidth fluctuations at the transport layer. Bandwidth and Buffer maintain lower but more consistent quality.

Stall time. Tabels I and II also capture the average number of stalls and stall duration across all three videos and for each video individually. Further, Fig 4(a), (b), and (c) show average stall duration plotted against average visual quality of experience (bitrate, VMAF, and PSNR, respectively). It is immediately apparent from Fig 4 that LoL$^+$ is an outlier with very high visual quality of experience but also very high stall time, up to 3-4 seconds on average, while Bandwidth and Buffer maintain lower qualities and stall time (consistently less than 1 second).

After examining our traces, we can conclude that most of this (still not all) is due to the startup behavior of each strategy, and in particular LoL$^+$'s optimistic startup that begins at very high quality, inducing potentially long stalls. Bandwidth and Buffer both have much more pessimistic startup phases. Bandwidth attempts to exactly match available bandwidth resources and Buffer explicitly starts at the lowest quality.

Findings: LoL$^+$ uses an optimistic startup phase which leads it to trade off visual quality of experience for stall time. In contrast, Bandwidth and Buffer both make the opposite trade, starting at much lower bitrates (in some cases the lowest available) to avoid stalls and maintaining lower visual quality throughout the session.

Quality switches. Fig. 4(d) shows the average number of quality switches for each ABR mechanism as a function of channel condition. All three strategies show a large number of quality switches in the mobile case as they attempt to track the highly variable link quality. Interestingly, LoL$^+$ is least prone to switch qualities in the stationary channel conditions but much worse in the Mobile case. While the number of quality switches used by Buffer and Bandwidth increase by factors of 2 and 3 from stationary to mobile scenarios, the number of quality switches LoL$^+$ uses increases by a factor of 4.

Findings: The tradeoff between the ability to avoid stalls and the ability to take advantage of changing wireless throughput is a common theme across adaptive bitrate literature. In mmWave wireless, the changes in throughput can be quite extreme, and quality switches large. Strategies which do not take advantage of these sudden changes may not take advantage of higher rates, but ones which do cause frequent switches.

VI. Conclusion

Highly dynamic mmWave wireless links and ultra high data rate requirements of lifelike VR applications pose new

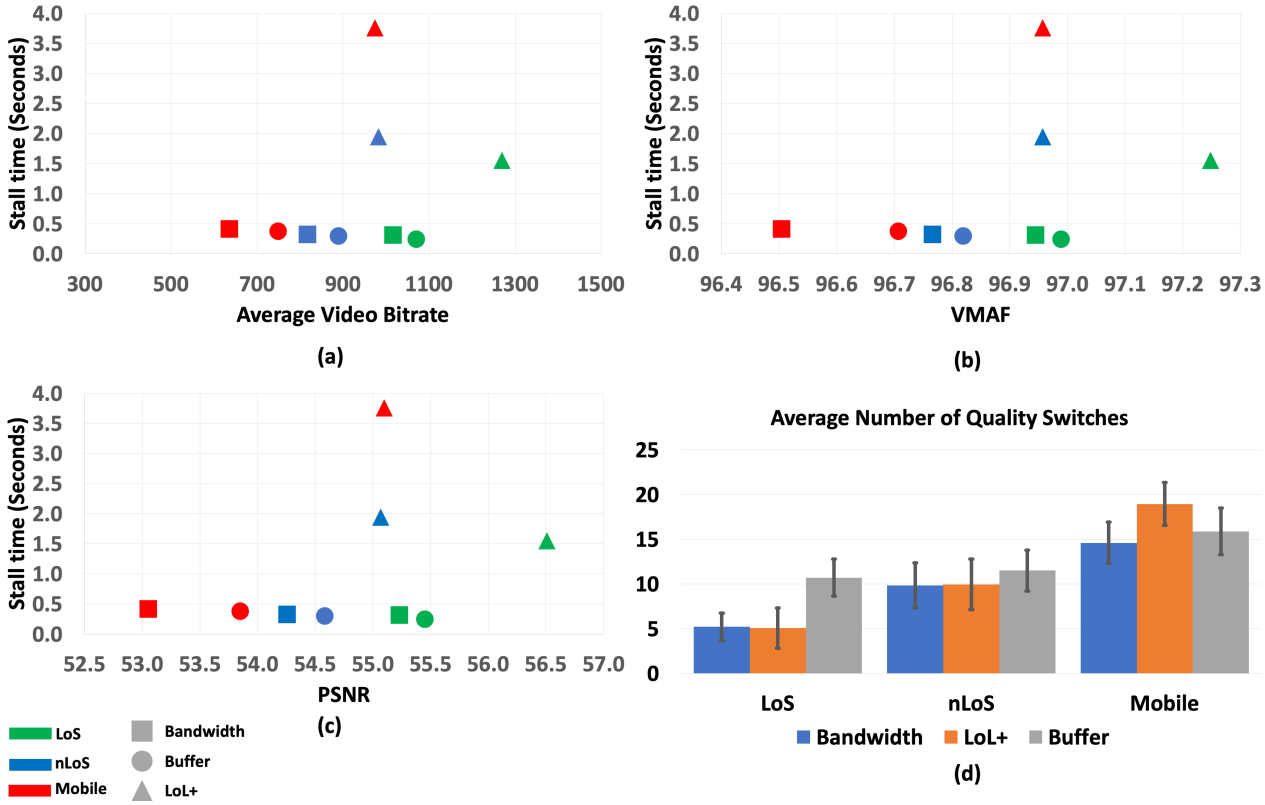

Fig. 4. **(a-c): Stall time vs visual quality of experience for 3 different quality of experience metrics (bitrate, PSNR, and VMAF, respectively). Each plot shows the results for all three ABR mechanisms. Lol+ has high stall durations, particularly in the startup phase due to optimistic initial settings. (d): Number of quality switches by each strategy across each channel condition. The error bars represent the standard deviation. LoL$^+$ is least prone to switch qualities in the best case (stationary LoS/nLoS) but much worse in the worst case (mobile).**

challenges for state-of-the-art ABR video streaming methods. We empirically evaluate the performance of three such ABR methods in three different mmWave network conditions for several representative 360° videos. We show that ABR methods that aim to simultaneously increase quality of experience and reduce stall time, generally do not achieve both objectives. In particular, increasing visual quality can lead to a tenfold increase in stall time, while decreasing stall time can lead to a 34% drop in video bitrate. As future work, we plan to investigate additional ABR strategies across a broader variety of videos and higher number of VR clients engaged in longer and more complex interactive scenarios.

VII. Acknowledgements

This work was supported in part by the NSF under awards CNS-1942305, CNS-1910517, CCF-2031881, ECCS-2032387, CNS-2040088, CNS-2032033, and CNS-2106150; by the NIH under award R01EY030470; and by the Panasonic Chair of Sustainability at NJIT.

References

[1] 3GPP, "Study on localized mobile metaverse services," in *5G Release 19*, 2022.

[2] GSMA, "Cloud AR/VR whitepaper," https://www.gsma.com/futurenetworks/wiki/cloud-ar-vr-whitepaper/.

[3] "Internet connection speed recommendations," https://help.netflix.com/en/node/306.

[4] J. Chakareski, M. Khan, and M. Yuksel, "Towards enabling next generation societal virtual reality applications for virtual human teleportation," *IEEE Signal Process. Mag.*, vol. 39, no. 5, September 2022.

[5] M.-L. Champel, T. Stockhammer, T. Fautier, E. Thomas, and R. Koenen, "Quality requirements for VR," in *Proc. 116th MPEG meeting ISO/IEC JTC1/SC29/WG11 (MPEG)*, 2016.

[6] "An evaluation of AT&T millimeter wave markets," https://www.allnetinsights.com/blogs/news/att-millimeter-wave-markets.

[7] M. Gapeyenko, A. Samuylov, M. Gerasimenko, D. Moltchanov, S. Singh, M. R. Akdeniz, E. Aryafar, S. Andreev, N. Himayat, and Y. Koucheryavy, "Spatially-consistent human body blockage modeling: a state generation procedure," *IEEE Trans. Mobile Computing*, 2020.

[8] C. G. Ruiz, A. Pascual-Iserte, and O. Munoz, "Analysis of blocking in mmwave cellular systems: Application to relay positioning," *IEEE Trans. Communications*, vol. 69, no. 2, February 2021.

[9] A. Ansari and M. Wang, "iStream Player: A versatile video player framework," in *Proc. 33rd Workshop Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, June 2023.

[10] A. Bentaleb, M. N. Akcay, M. Lim, A. C. Begen, and R. Zimmermann, "Catching the moment with LoL$^+$ in Twitch-like low-latency live streaming platforms," *IEEE Trans. Multimedia*, February 2021.

[11] 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," in *Proc. 2014 ACM Conf. on SIGCOMM*, 2014.

[12] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "BOLA: Near-optimal bitrate adaptation for online videos," *IEEE/ACM Trans. Networking*, vol. 28, no. 4, August 2020.

[13] K. Spiteri, R. Sitaraman, and D. Sparacio, "From theory to practice: Improving bitrate adaptation in the DASH reference player," *ACM Trans. Multimedia Computing, Communications, and Applications*, 2019.

[14] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with FESTIVE," *IEEE/ACM Trans. Networking*, vol. 22, no. 1, February 2014.

[15] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. C. Begen, and D. Oran, "Probe and adapt: Rate adaptation for HTTP video streaming at scale," *IEEE Journal on Selected Areas in Communications*, April 2014.

[16] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over HTTP," in *Proc. ACM SIGCOMM*, August 2015.

[17] T. Karagkioules, R. Mekuria, D. Griffioen, and A. Wagenaar, "Online learning for low-latency adaptive streaming," in *Proc. ACM Multimedia Systems Conf.*, 2020.

[18] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proc. Conf. ACM Special Interest Group on Data Communication (SIGCOMM)*, August 2017.

[19] J. Struye, H. K. Ravuri, H. Assasa, C. Fiandrino, F. Lemic, J. Widmer, J. Famaey, and M. T. Vega, "Opportunities and challenges for virtual reality streaming over millimeter-wave: An experimental analysis," in *Proc. 13th Int. Conf. Network of the Future (NoF)*, 2022.

[20] T.-T. Le, D. N. Van, and E.-S. Ryu, "Real-time 360-degree video streaming over millimeter wave communication," in *Proc. Int. Conf. on Information Networking (ICOIN)*, 2018.

[21] J. Chakareski, M. Khan, T. Ropitault, and S. Blandino, "Millimeter wave and free-space-optics for future dual-connectivity 6DOF mobile multi-user VR streaming," *ACM Trans. Multimedia Computing Communications and Applications*, vol. 19, no. 2, pp. 57:1–25, Feb. 2023.

[22] S. Gupta, J. Chakareski, and P. Popovski, "Mmwave Networking and Edge Computing for Scalable 360° Video Multi-User Virtual Reality," *IEEE Trans. Image Processing*, 2023.

[23] S. Srinivasan, S. Shippey, E. Aryafar, and J. Chakareski, "FBDT: Forward and Backward Data Transmission Across RATs for High Quality Mobile 360-Degree Video VR Streaming," in *Proc. 14th ACM Conf. on Multimedia Systems (MMSYS)*, 2023.

[24] J. Chakareski and M. Khan, "Live 360° video streaming to heterogeneous clients in 5G networks," *IEEE Trans. Multimedia*, Apr. 2024.

[25] C. R. Storck and F. Duarte-Figueiredo, "A performance analysis of adaptive streaming algorithms in 5G vehicular communications in urban scenarios," in *Proc. IEEE Symp. Computers and Communications*, 2020.

[26] E. Ramadan, A. Narayanan, U. K. Dayalan, R. A. K. Fezeu, F. Qian, and Z.-L. Zhang, "Case for 5G-aware video streaming applications," in *Proc. Workshop 5G Measurements, Modeling, and Use Cases*, 2021.

[27] K. Arunruangsirilert, B. Wei, H. Song, and J. Katto, "Pensieve 5G: Implementation of RL-based ABR algorithm for UHD 4K/8K content delivery on commercial 5G SA/NR-DC network," in *Proc. Wireless Communications and Networking Conf. (WCNC)*, 2023.

[28] F. Y. Yan, H. Ayers, C. Zhu, S. Fouladi, J. Hong, K. Zhang, P. Levis, and K. Winstein, "Learning in situ: A randomized experiment in video streaming," in *Proc. 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2020.

[29] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Processing*, vol. 13, no. 4, 2004.

[30] A. Aaron, Z. Li, M. Manohara, J. Y. Lin, E. C.-H. Wu, and C.-C. J. Kuo, "Challenges in cloud based ingest and encoding for high quality streaming media," in *Proc. IEEE Int'l Conf. Image Processing*, 2015.

[31] J. Chakareski, J. Apostolopoulos, W.-T. Tan, S. Wee, and B. Girod, "Distortion chains for predicting the video distortion for general packet loss patterns," in *Proc. Int'l Conf. Acoustics, Speech, and Signal Processing*, vol. 5. Montreal, Canada: IEEE, May 2004, pp. 1001–1004.

[32] "DASH Reference Player," https://reference.dashif.org/dash.js/.

[33] "Real-Time Communication for the Web," https://webrtc.org/.

[34] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar, J. Bailey, J. Dorfman, J. Roskind, J. Kulik, P. Westin, R. Tenneti, R. Shade, R. Hamilton, V. Vasiliev, W.-T. Chang, and Z. Shi, "The QUIC transport protocol: Design and internet-scale deployment," in *Proc. ACM Special Interest Group on Data Communication (SIGCOMM)*, 2017.

[35] X. Liu, Y. Huang, L. Song, R. Xie, and X. Yang, "The SJTU UHD 360-Degree immersive video sequence dataset," in *Proc. 2017 International Conf. Virtual Reality and Visualization (ICVRV)*, 2017.

[36] M. Lim, M. N. Akcay, A. Bentaleb, A. C. Begen, and R. Zimmermann, "When they go high, we go low: Low-latency live streaming in Dash.JS with LOL," in *Proc. ACM Multimedia Systems Conf.*, 2020.

TABLE II

RESULTS TABLE OF METRICS FOR THREE 360° VIDEOS IN THE EXPERIMENT. THE BEST RESULTS FOR A GIVEN CHANNEL CONDITION IN EACH VIDEO ARE BOLDED.

Academic 360° video				
Mobility		Stationary		Mobile
Line of Sight		LoS	nLoS	
Banwidth	VMAF	97.09	96.93	96.07
	PSNR	56.39	55.61	51.67
	Selected Bitrate	1028.10	860.68	256.09
	Number of Stalls	1.00	1.00	1.00
	Time of Stalls	0.26	**0.27**	0.57
	Log QoE	73.94	64.85	14.43
	Linear QoE	324.52	253.01	59.33
Buffer	VMAF	97.21	**96.98**	96.71
	PSNR	56.42	**55.70**	54.09
	Selected Bitrate	1048.81	**881.62**	601.04
	Number of Stalls	1.00	1.00	1.00
	Time of Stalls	**0.25**	0.34	**0.45**
	Log QoE	74.29	**66.72**	**41.08**
	Linear QoE	342.45	**272.48**	101.96
LoL+	VMAF	**97.53**	96.86	**97.07**
	PSNR	**57.40**	54.61	**55.71**
	Selected Bitrate	**1238.88**	687.14	**868.89**
	Number of Stalls	1.00	1.25	3.80
	Time of Stalls	1.28	2.16	5.13
	Log QoE	**80.87**	49.26	37.68
	Linear QoE	**400.09**	167.52	**140.63**
Runner 360° video				
Mobility		Stationary		Mobile
Line of Sight		LoS	nLoS	
Banwidth	VMAF	96.81	96.55	96.61
	PSNR	55.15	53.80	54.11
	Selected Bitrate	1010.08	760.72	817.98
	Number of Stalls	1.00	1.00	1.05
	Time of Stalls	0.32	0.35	**0.33**
	Log QoE	75.53	63.11	59.34
	Linear QoE	339.80	236.49	202.54
Buffer	VMAF	96.81	96.59	96.59
	PSNR	55.47	54.06	54.08
	Selected Bitrate	1080.23	816.49	813.31
	Number of Stalls	1.00	1.00	1.00
	Time of Stalls	**0.23**	**0.28**	**0.33**
	Log QoE	73.66	63.65	**61.42**
	Linear QoE	330.93	237.85	220.91
LoL+	VMAF	**97.10**	**96.95**	**96.87**
	PSNR	**56.75**	**55.71**	**55.25**
	Selected Bitrate	**1309.41**	**1107.60**	**1020.01**
	Number of Stalls	1.00	1.10	1.80
	Time of Stalls	1.59	1.80	3.40
	Log QoE	**81.96**	**71.35**	56.03
	Linear QoE	**423.06**	**315.63**	**226.21**
Kaimenxueye (Gate) 360° video				
Mobility		Stationary		Mobile
Line of Sight		LoS	nLoS	
Banwidth	VMAF	96.94	96.82	96.83
	PSNR	54.17	53.37	53.39
	Selected Bitrate	1012.83	833.15	834.73
	Number of Stalls	1.00	1.00	1.00
	Time of Stalls	0.36	0.35	**0.33**
	Log QoE	75.63	65.37	61.01
	Linear QoE	340.53	251.65	212.38
Buffer	VMAF	96.94	96.89	96.82
	PSNR	54.47	54.00	53.39
	Selected Bitrate	1083.95	977.47	835.58
	Number of Stalls	1.0	1.0	1.0
	Time of Stalls	**0.25**	**0.25**	**0.33**
	Log QoE	74.39	70.20	**62.38**
	Linear QoE	338.88	**295.65**	227.56
LoL+	VMAF	**97.11**	**97.06**	**96.99**
	PSNR	**55.38**	**54.89**	**54.35**
	Selected Bitrate	**1261.72**	**1153.63**	**1035.50**
	Number of Stalls	1.0	1.11	1.75
	Time of Stalls	1.78	1.86	2.73
	Log QoE	**78.53**	**73.11**	60.60
	Linear QoE	**397.41**	243.88	**333.17**