PRISM-XR: Predictive Real-Time Intelligent Streaming for 360-Degree Multimedia

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

***Abstract*—New, bandwidth intensive video formats intended for Metaverse content such as virtual and augmented reality threaten to strain radio resources beyond the breaking point. To address these challenges, both the cellular and WiFi standardization communities have adopted mmWave wireless to provide ultra-high data rate and low-latency communication. However, mmWave systems are susceptible to blockages, which can cause sudden and wide fluctuations in throughput, which Adaptive Bitrate (ABR) strategies are too slow to react to. This paper introduces a proactive video bitrate adaptation system called PRISM-XR, which builds on recent advances in wireless research predicting and mitigating blockages at the physical and link layers. PRISM-XR operates in tandem with other ABR strategies, interrupting control when notified of a blockage by predictive intelligence available at the physical layer. In order to characterize video streaming over mmWave, we carry out Over-The-Air (OTA) experiments using real mmWave radios. Our evaluation highlights the role of the transport layer (e.g., TCP vs QUIC) in video quality, finding that the use of QUIC results in severe performance degradation. We then evaluate our solution through Mininet emulations, and show that PRISM-XR can significantly improve QoE (Quality-of-Experience) over standard ABR methods when using TCP as the transport protocol.**

I. Introduction

HTTP Adaptive Streaming (HAS), standardized as Dynamic Adaptive Streaming over HTTP (DASH) and HTTP Live Streaming (HLS), is a widely adopted technology for delivering high-quality videos while coping with variable data rates. The operation is relatively simple and can be cleanly divided into two components: (i) A server which provides videos encoded at multiple bitrates; and (ii) A client which downloads parts of the video in sequence, the bitrate being decided by an Adaptive Bitrate (ABR) strategy so as to maximize the viewer's Quality of Experience (QoE) by matching the network conditions. HAS enjoys wide popularity, being used in both video on demand services (e.g., YouTube) as well as low-latency live streaming (e.g., Twitch) where only a few seconds of latency are tolerated.

It would be natural to distribute Metaverse content, such as 360° or point-cloud videos, through HAS as well. But these new forms of video require extremely high resolutions and frame rates to deliver high quality, immersive experiences [1], which will require large improvements in wireless data rates. This in turn necessitates the use of emerging technologies at the physical layer, such as millimeter wave (mmWave), which can deliver several 100s of Mbps to multi-Gbps speeds. MmWave systems achieve these rates by forming highly directional beams and leveraging higher wireless bandwidth by operating in higher frequency ranges. MmWave technology is already employed by both cellular and WiFi operators as part of 5G and 802.11 ad (also known as WiGig), respectively.

However, mmWave frequencies are inherently susceptible to blockages, e.g., human body alone can block the signal by more than 20 dB [2]. As a result, mmWave systems can suffer large fluctuations in data rate. The wireless community has begun to address this issue by developing blockage prediction and mitigation techniques, for instance predicting a blockage in the next few seconds and proactively handing the client off to another access point (AP) [2] or adapting the beams at the AP, client, or both [3]. While these methods achieve remarkable success in preventing complete blockage, they are often unable to prevent large fluctuations in data rate: An AP which does not suffer from the blockage may be further away from the client, or out of the client's Line-of-Sight (LoS). Since ABR strategies are receiver-driven, there is often significant lag in addressing these fluctuations.

Our goal in this paper is to improve a streaming client's QoE in mmWave networks by making use of predictive intelligence available at the physical layer. To accomplish this, we begin by using commercial-off-the-shelf (COTS) mmWave equipment to run over-the-air (OTA) experiments to characterize the performance of existing ABR strategies over mmWave links under diverse channel conditions. We also run our experiments using QUIC, and show the outsized influence the choice of transport layer protocol has on video QoE over the last-hop links. We then show how to achieve better video streaming performance in mmWave systems by designing a system, called PRISM-XR, which briefly interrupts another ABR strategy on receipt of a blockage warning and creates a download plan to handle the blockage. We make the following contributions:

- **System Development.** We develop an integrated hardware–software system to evaluate both PRISM-XR and other ABR strategies under controlled network conditions, with support for multiple transport protocols, including TCP and QUIC.
- **Over-The-Air (OTA) Evaluation of ABR Streaming with QUIC and TCP over WiGig.** We perform a first-of-its-kind comparison of TCP and QUIC with ABR and OTA streaming in mmWave settings with LoS, non-Line-of-Sight (nLoS), and mobile clients. We show that under the same networking conditions, QUIC and TCP create almost separate clusters in terms of stalls, bitrate, and PSNR, with QUIC generally resulting in bitrates that are 30%-50% lower than those of TCP.
- **PRISM-XR Evaluation.** We evaluate the performance

of PRISM-XR used alongside several existing ABR schemes under different blockage models. We show that PRISM-XR can provide 10%-40% gains in terms of QoE against un-augmented ABR strategies when used with TCP. However, gains diminish in the presence of errors in the predictive intelligence system as well as errors introduced by QUIC's more variable throughput. We perform detailed analysis to quantify these errors.

The paper is organized as follows. We discuss the related work in Section II. We present our system model, problem formulation, and design of PRISM-XR in Section III. We present the results of our extensive performance evaluations in Section IV. Finally, we conclude the paper in Section V.

II. Related Work

Proactive Blockage Prediction and Mitigation. The wireless community has developed many methods to address the challenges posed by blockages. Many works have presented systems which can accurately predict many aspects of upcoming blockages, such as onset, duration, and impact on the client's data rate [3], [4], [5], [6], [7]. Predicted blockages can also be mitigated using a variety of methods, such as beam switching or handoff [3]. Of course, no mitigation technique can completely eliminate the impact of a blockage and steep changes in throughput can still result from the mitigation itself. PRISM-XR builds on top of existing work on physical layer based blockage prediction methods to adapt the video download plan in a manner that optimizes the client QoE.

Adaptive Bitrate Strategies. Many methods have been developed to optimize bitrate selection in video streaming, aiming to balance video quality, stalls (re-buffering), and latency. These approaches generally fall into four categories based on what information they use to make decisions: (i) Buffer-based strategies like BBA [8], which rely on buffer level; (ii) Bandwidth-estimate based approaches [9], which estimate the underlying bandwidth; (iii) Control-theoretic and online-learning strategies, such as FastMPC [10] and LoL$^+$ [11]; and (iv) Reinforcement-learning based strategies such as Pensieve [12]. Performance of some of these ABR strategies have been evaluated over mmWave wireless links, e.g., through small bandwidth throughput traces in [13] or high bandwidth measurements in [14]. However, none of these works evaluate the impact of transport protocol choice or propose a solution to proactively mitigate blockages at the application layer.

QUIC Evaluation. It has been observed that QUIC suffers severe performance degradation compared to TCP/TLS in high-speed wired testbeds, with throughput declining up to 40% [15], [16], largely due to excessive receiver-side processing overhead. However, what impact this has on high-bandwidth wireless is underexplored. In this paper, we present first-of-its-kind OTA QUIC measurements over Gbps wireless links and show that ABR schemes under QUIC select bitrates that are on average 30%-50% lower than similar ABR schemes under TCP. However, stall time is mostly unaffected by the choice of the transport protocol.

III. ABR Streaming with Predictive Intelligence

In this section, we present our system model, define our QoE maximization problem, and present the details of PRISM-XR.

A. System Model

We begin by describing the dynamics of a video player followed by the dynamics of the mmWave link.

Video Player Model. A video streaming session includes a client, called a player, and a server from which the video is downloaded in pieces called *segments*, each L seconds long. We assume that the full video is composed of K segments, and each segment is encoded in advance using different encoding parameters (e.g., bitrates) from which the client may choose to download each segment from. We use R_k to denote the selected bitrate of the kth segment in the client's buffer. We use t to denote time and t_k to denote the time the kth element finishes downloading. Each R_k is drawn from a set of representations $\mathcal{R}$. We use $d(R_k)$ to denote the file size of the kth segment. We assume constant birtrate (CBR) encoding method for video compression, and hence $d(R_k)$ is equal to the bitrate (R_k) times the video segment duration (L). Further, we assume the existence of a non-decreasing function $q(R_k)$ corresponding to the segment's visual quality.

The video player's buffer is assumed to be constrained (by memory or distance to live edge) to B_{max} seconds, and cannot fall below zero. By distance to live edge, we mean the time difference between when a new segment is available on the server and when started to be viewed by the client. Let C denote the mean data rate (or transport layer throughput) and let's assume the buffer level is at least L seconds below B_{max}. Then, the buffer state at time t_k (i.e., immediately after downloading segment k) and any other time $t + \Delta t$ such that $t_k < t + \Delta t < t_{k+1}$ is:

$$B(t_k) = \max\left(0, B(t_{k-1}) - \frac{d(R_k)}{C}\right) + L, \tag{1}$$

$$B(t + \Delta t) = \max\left(0, B(t) - \Delta t\right), \tag{2}$$

where the max ensures that the buffer level is non-negative.

Network Model. We consider a network setup with fixed clients and access points (APs), and mobile blockers. The APs and clients operate over a mmWave band (e.g., WiGig). We assume each client and its serving AP are initially in LoS channel condition with respect to one another, which results in a stable link condition with mean data rate of C_i. The LoS or nLoS channel condition can be accurately predicted in mmWave systems by leveraging the underlying in-band communication without incurring any additional communication overhead (e.g., [17]).

We assume an underlying physical layer blockage prediction method (e.g., [4]) that provides an accurate prediction of blockage instant, the duration of the blockage, and change in the data rate[1]. The underlying prediction method may additionally employ a blockage mitigation technique to proactively

[1]We use the terms throughput, bandwidth, and data rate interchangeably.

mitigate the blockage, e.g., by handing off the client to another AP. We use T_1 to denote the time instant in which the physical layer module has provided the application layer an advanced warning about a predicted (or scheduled[2]) sudden change in data rate. We use C_1 to denote the mean data rate prior to the blockage event. We use T_2 to denote the time instant of change in the data rate (e.g., due to blockage happening or change in the serving AP) and C_2 the predicted new mean throughput. We use T_3 to denote the end of the blockage duration. At this point, the underlying communication may switch to a new mean data rate C_3. For example, if the serving AP was changed to mitigate the blockage, the client may revert to its previous serving AP. We use T_4 to denote the end of the prediction-based time horizon provided by the physical layer. Fig. 1 provides a graphical representation of these events.

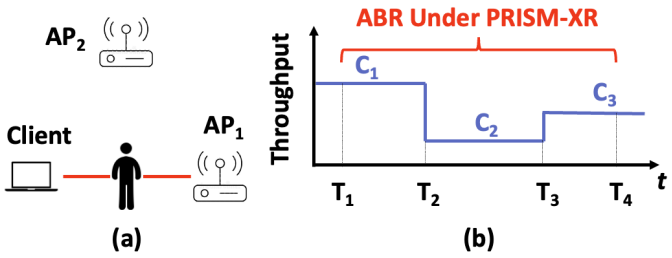

Fig. 1. **(a): Client initially has a LoS channel to its serving AP (AP_1). Upon detection of an upcoming blocker, client proactively switches to an alternative LoS AP (AP_2). (b): At time T_1 the radio predictive module informs the client application of scheduled sudden changes in throughput at times T_2 and T_3, with a time horizon until T_4. The predicted data rates are C_2 and C_3, respectively. PRISM-XR takes over ABR adaptation from T_1 to T_4. ABR before T_1 and after T_4 is under control of an existing ABR mechanism.**

B. Problem Formulation

The goal of a download plan at the client is to select the number of segments to download and the bitrate for each segment such that the QoE at the client is maximized. With abuse of notation, let $R_1, \ldots, R_N$ denote the selected bitrates of the next N segments once the client's application receives notification of upcoming sudden changes in throughput at time T_1. Our goal is to maximize QoE by maximizing the sum of visual quality across all segments, while penalizing quality changes across consecutive segments as well as stall (re-buffering) events. Specifically:

[2]We use the term *scheduled*, since the mitigation technique may decide to proactively take an action (e.g., do handoff) in anticipation of blockage. Further, it is possible for the notification to occur in the middle of a segment download. However, our low-latency setting uses very short (0.5 second) segments and a relatively shallow maximum buffer size (3 seconds), so the difference between T_1 and the start of the first download using was always very small. With larger segments or a deeper maximum buffer, it may be necessary to interrupt the current download instead of simply waiting as the remaining download time could be significant.

$$\mathcal{P}_1: \max_{\{R_1,\ldots,R_N,N\}} QoE = w_1 \sum_{k=1}^{N} q(R_k) - w_2 \sum_{k=1}^{N} |q(R_k) - q(R_{k-1})|^P - w_3 \sum_{k=1}^{N} \max\left(0, \frac{d(R_k)}{C} - B(t_{k-1})\right)$$

$$s.t. \sum_{k=1}^{N} d(R_k) \leq C_1(T_2 - T_1) + C_2(T_3 - T_2) + C_3(T_4 - T_3)$$

$$B(t) \leq B_{\max}, \forall t \in [T_1, T_4]$$

here w_1 represents the weight of visual quality, w_2 weight of quality variation, w_3 weight of stall times, and P the exponentiation term in quality switches. All of these are input variables and may be configured by the video player. The first constraint in problem $\mathcal{P}_1$ ensures that the total size of selected files can be supported by the system bandwidth. The second constraint ensures that at anytime the video player buffer is below the maximum value. The output of the optimization problem are the number of segments (N) to download along with the appropriate bitrates for each segment.

Problem $\mathcal{P}_1$ is non-convex due to the quality switches term in the QoE. Further, even for a fixed N, exhaustive search may be infeasible. For example, consider a scenario where mmWave data rates range from tens of Mbps to a few Gbps, with 20 different bitrate representations per video segment, each lasting 0.5 seconds, and a time horizon of 10 seconds. Thus, exhaustive search would need to iterate over 20^{20} different combinations, which is impractical even with the fastest processors available today. Thus, we need to resort to heuristics to solve the problem in $\mathcal{P}_1$.

C. Design of PRISM-XR

In this section, we discuss the design of PRISM-XR, a greedy algorithm developed to solve problem $\mathcal{P}_1$. We use the following features in the design of PRISM-XR: (i) The player has knowledge of future average throughput values and their evolution over time. (ii) The player is aware of the bitrate ladder at the server, enabling the computation of file sizes for future video segment representations. (iii) The player is cognizant of the current video buffer status.

PRISM-XR emulates the passage of time to ensure that the selected video segments fit within the buffer. Specifically, if the current buffer level cannot accommodate a new segment, PRISM-XR delays downloading until enough space becomes available. It is important to note that all segments occupy the same duration in the buffer (L), regardless of their bitrate.

Furthermore, PRISM-XR operates in a greedy manner. It starts by selecting the bitrate for the first segment to be downloaded. Next, it determines the bitrate for the second segment based on the choice made for the first segment. This iterative process continues, enabling PRISM-XR to identify both the

number of segments and their bitrates to be downloaded within the time interval T_1 to T_4.

Let function f represent the bandwidth function shown in Fig. 1(b). Let R_0 denote the bitrate of the last segment downloaded before PRISM-XR takes over ABR operations. Let's assume the buffer level can accommodate a new segment. Then, PRISM-XR selects the bitrate of the first segment as:

$$R_1^* = \text{argmax}_{R \in \mathcal{R}} QoE \quad s.t.\ d(R) \leq \int_{T_1}^{T_4} f_t\, dt \tag{3}$$

Here, QoE is equal to:

$$w_1 q(R_k) - w_2 |q(R_k) - q(R_{k-1})|^P - w_3 \max\left(0, \frac{d(R_k)}{C} - B(t_{k-1})\right) \tag{4}$$

with k equal to one. In other words, we select the bitrate for the first segment as the one that maximizes the QoE expression in Eq. (4), subject to the constraint that the file size can be supported by the system bandwidth. To determine the bitrate of the second segment, we first find the time instant (t') that the first download finishes. Next, we select the bitrate of the second segment as:

$$R_2^* = \text{argmax}_{R \in \mathcal{R}} QoE \quad s.t.\ d(R) \leq \int_{t'}^{T_4} f_t\, dt \tag{5}$$

here we reuse the QoE expression in Eq. (4) (with $k = 2$).

Algorithm 1 outlines the details of PRISM-XR. First, if the buffer cannot accommodate a new segment, PRISM-XR simulates a pause in time by shifting the time variable (t) to allow space for the next segment (Lines 3–5). Next, it selects the bitrate that maximizes QoE (Line 7) and calculates the time instant at which the segment download would complete (Line 8). Since the bandwidth function f is a piecewise linear step function, this calculation can be performed in closed form. At the end, the algorithm returns the number and bitrate of segments to be downloaded.

Algorithm 1 PRISM-XR: Predictive Real-time Intelligent Streaming for 360° Multimedia

1: $t \leftarrow T_1$;
2: **for** i = 1 to ∞ **do**
3: **if** $B(t) + L > B_{\max}$ **then**
4: $t = t + B(t) - (B_{\max} - L)$
5: **end if**
6: **if** t ¡ T_4 **then**
7: $R_i^* = \text{argmax}_{R \in \mathcal{R}} QoE$ s.t. $d(R) \leq \int_t^{T_4} f_t\, dt$
8: find new time (t') by solving $t' = \frac{d(R_i^*)}{\int_t^{t'} f_t\, dt}$
9: $t = t'$
10: **else**
11: Break
12: **end if**
13: **end for**
14: **return** i, $R_1^*, \ldots, R_i^*$

IV. Performance Evaluation

In this section, we conduct OTA experiments with COTS hardware using iStream Player [18] to quantify and analyze the performance of three existing ABR strategies, while also documenting their behavior when tunneling over QUIC. We then perform emulated experiments using Mininet [19] to evaluate the same strategies augmented with PRISM-XR. We resort to emulation because our mmWave hardware does not give us the kind of low-level access necessary to implement blockage prediction as described in the literature.

A. OTA Experimental Setup

We begin by performing OTA experiments using ultra-high-bitrate videos to better understand the performance of existing ABR strategies over mmWave wireless links. We employ a customized version of iStream Player with support for an additional ABR strategy along with additional modifications to support tunneling connections over quic-go [20]. We intend to make our software and dataset public upon publication.

Network Deployment. We set up our network using a NetGear NightHawk X10 router (AP) communicating with a TravelMate laptop (client), using a desktop PC as an edge server and a TurtleBot robot to standardize movement of the client in mobile scenarios. These devices are shown in Fig. 2(a) and are deployed in an indoor office environment.

Video Dataset. We use three 8K 360°videos selected randomly from the SJTU 8K VR dataset [21]. We set the segment size to 1 second and used 9 profiles, setting bitrates at 150, 250, 450, 550, 650, 900, 1200, and 1500 Mbps.

Video Player. We use iStream Player [18] as the video player. iStream Player offers extensive metrics and detailed logs, making analysis easier. This functionality helps to isolate performance issues, e.g., attributing them to choice of transport layer protocol rather than other components like the web browser, video player, or web server. During all streaming sessions, the buffer size is capped at 3 seconds.

QUIC Implementation. iStream Player natively supports QUIC through Python's aioquic library, and we initially planned to evaluate QUIC using this built-in module. However, preliminary tests with the built-in QUIC downloader yielded much worse results compared to TCP. In many cases, iStream Player with QUIC struggled to sustain even the lowest bitrate profile in our dataset. Given that QUIC implementations exhibit significant variations in maximum achievable throughput [16], we decided to explore alternatives. We chose to tunnel iStream Player through quictun [22], an open-source QUIC tunnel implemented using quic-go. While the peak throughput with quictun is still lower than TCP — about 600 Mbps compared to 1300 Mbps over TCP — its performance is similar to Fastly's quicly (which we measured at about 600-700 Mbps) and better than the Rust library quinn (100-200 Mbps) and aioquic (25-100 Mbps).

Mobility Patterns. To examine the impact of different channel conditions on video streaming performance, we employ three distinct mobility and channel condition scenarios. These scenarios are illustrated in Fig. 2(b) and include:

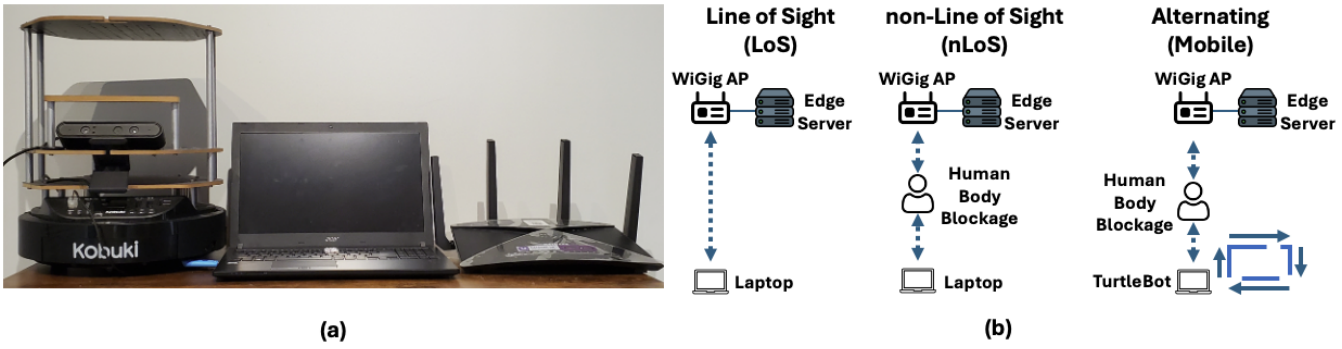

Fig. 2. **(a): Our hardware includes a TurtleBot robot, an Acer TravelMate laptop, and a NetGear NightHawk X10 router. (b): Diagram showing the 3 mobility and channel condition scenarios: static LoS, static nLoS, and a mobile pattern alternating between LoS and nLoS. In all cases, the laptop begins from 2 meters away from the AP. In nLoS and mobile experiments, a human stands between the AP and the laptop, 1 m away from the router. In the mobile scenario, the client traverses a 2x1 m^2 rectangle.**

1) **LoS:** The client is placed 2 meters away from the AP and remains stationary for the duration of the experiment. No blockages are introduced.
2) **nLoS:** A human body blockage is placed 1 meter away from the AP. Both the client and blocker remain stationary during the experiment. The throughput in the nLoS scenario is lower than the throughput in the LoS scenario.
3) **Mobile:** The client is attached to a TurtleBot robot, which moves in a 2x1 m^2 rectangle over several seconds. A human body blockage is introduced 1 m between the AP and the starting point of the client's mobility pattern. The blocker remains stationary. The client abruptly changes between LoS and nLoS, producing a variable throughput.

ABR strategies. Many ABR strategies have been developed to tame throughput instability in both wireless and wired settings. We evaluate a bandwidth heuristic (labeled BW in result figures, configured to use 70% of bandwidth estimated over a 10s sliding window), a buffer-based algorithm (labeled BBA [8], configured to always keep a 1s "reservoir" of content in the buffer even at the expense of quality, with a 2s "upper reservoir" in which quality improves linearly), and LoL$^+$ [11].

B. Results of OTA Adaptive Streaming over WiGig

We perform 40 trials (20 with TCP, 20 with QUIC) over each combination of video, ABR strategy, and channel conditions. In this section, we analyze the results of our OTA experiments. We discovered a large change in achievable video bitrate as a result of choice of transport layer.

Stall Time and Visual Quality. Fig. 3(a) and (b) show the relationship between the average stall time and two measures of visual quality: average video bitrate (Fig. 3(a) in Mbps) and average PSNR (Fig. 3(b) in dB). In both representations, tests performed using TCP and those performed using QUIC form largely separate clusters, overlapping only slightly in terms of PSNR. Using QUIC always results in a few dB lower PSNR and 30%-50% lower bitrate compared to TCP. However, stall time is mostly unaffected by the choice of transport protocol except for LoL+ in the mobile scenario, where QUIC increases the average stall time by 12 seconds.

QUIC achieving lower throughput than TCP at high bandwidths has also been observed in [15], [16], though we are the first to evaluate its impact on video streaming at high bandwidths. Previous research [15] identifies the primary bottleneck as the lack of receiver-side offloading, which results in a higher number of QUIC packets being processed on the CPU rather than the network interface card. We also observed significantly higher CPU usage during QUIC video streaming sessions compared to TCP, supporting the same conclusion.

Visual Quality Switches. In Fig. 3(c), we show the average number of quality switches across trials. The results show that the impact of QUIC on session stability—whether it makes the session more or less unstable—depends on the ABR strategy and the specific channel conditions. Using Bandwidth (BW ABR) with QUIC results in increased quality switches in the nLoS scenario and decreased switches elsewhere, while using BBA and LoL$^+$ results in the reverse. Our analysis of the traces reveals that this effect stems from QUIC's tendency to amplify small variations in data rate by introducing additional delays. As a result, the Bandwidth strategy struggles to track minor fluctuations common in the nLoS scenario, while the other two strategies experience buffer level fluctuations despite no changes in the underlying data rate.

C. Emulator Setup for PRISM-XR Experiments

In this section, we describe the emulation setup we use to evaluate PRISM-XR. Since our hardware does not grant access to the low-level components necessary to actuallly implement phyiscal layer predictive intelligence, we resort to emulating mmWave environments using Mininet [19], which allows us to dynamically control links while accurately notifying the client in advance of impending changes.

Emulation Setup. We simulate a radio-level throughput predictor by emulating a network in software, controlling its data rate, simultaneously sending throughput predictions to higher layers as we manipulate the link. This process requires a network emulator and a modified video player.

Network Emulator. We use Mininet, a lightweight network emulator which uses namespace virtualization to simulate multiple hosts (including customization switches) on a network.

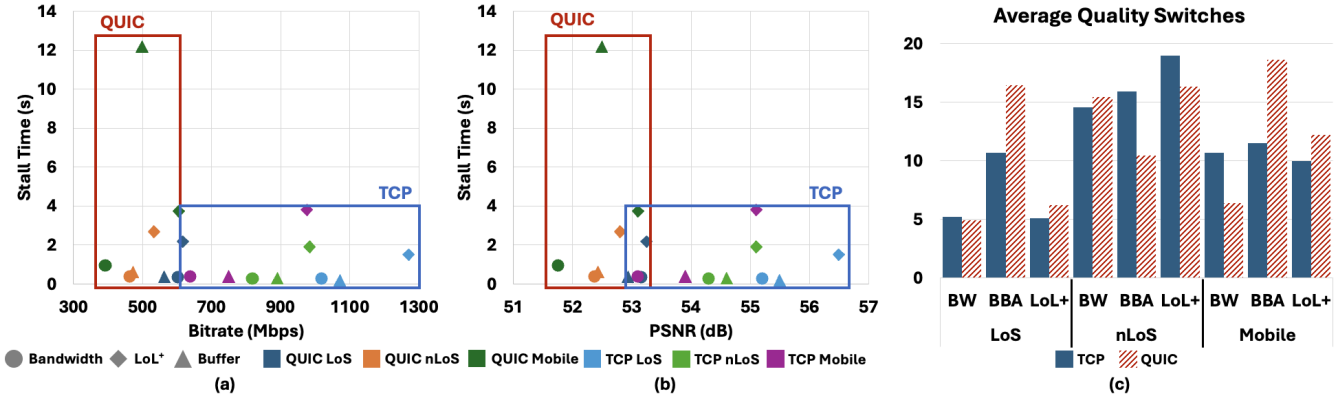

Fig. 3. **(a,b): Average stall time vs. average selected segment bitrate (a) and PSNR (b) with OTA experiments. TCP consistently achieves higher bitrates and visual quality (PSNR). However, stall time is mostly unaffected by the choice of the transport protocol, except for LoL+ in the mobile scenario. (c): Average number of quality switches across different ABR strategies and channel/mobility conditions.**

Each host is provided with its own IP address and port space while globally sharing a process space with the host machine. While fully simulating machines communicating over a network with containers or virtual machines was an option, Mininet proved much lighter weight, allowing us to iterate on experiments and solutions much faster. Within Mininet we simulate an edge server communicating with a client over a single intermediate switch. To simulate rate changes, we use Linux TC on the downlink interface from switch to client.

iStream Player. iStream Player required modifications to respond to network events in real time. To simulate advance warnings, we send messages through ZMQ, a lightweight, cross-platform IPC method. Upon receipt of one of these warnings, iStream Player then invokes PRISM-XR and generates a download plan as specified in Section III.

Video Dataset. We used four videos randomly selected from the same dataset described in Section IV-A. We changed our encoding parameters to better accommodate the limits of the emulation setup described above, namely the bandwidth limit resulting from placing both the client and server on the same physical machine. The videos were encoded using target bitrates of 20, 40, 80, 160, 320, and 640 Mbps, with a segment size of 0.5 seconds chosen to suit low-latency live streaming scenarios. Each video is 36 seconds long, comprising 72 segments. The videos exhibit diverse rate-distortion characteristics, particularly in segment file size variability.

QoE Parameters. In all experiments, the parameters of the QoE function from problem $\mathcal{P}_1$ (Section III-B) are fixed at $w_1 = 1$, $w_2 = 5$, $w_3 = 20$, and $P = 2$. [3] For the quality function q, we use a logarithmic model to capture the diminishing returns of encoding at higher target bitrates.

Scenarios. Our evaluation considers two scenarios characterized by different blockage or blockage mitigation behaviors, referred to as Scenario 1 (Fig. 4(a)) and Scenario 2 (Fig. 4(d)). Both scenarios assume that a blockage predictor can accurately estimate future throughput values. For convenience, we name the time intervals over which ABR decisions are under PRISM-XR's control as follows: The time from T_1 to T_2 is referred to as the Advance Interval, the time from T_2 to T_3 as the Blockage Interval, and the time from T_3 to T_4 as the End of Horizon (EoH) Interval. In both scenarios, we set the maximum buffer to 3 seconds, the Advance Interval to 3 seconds, and the EOH Interval to 6 seconds. The blocker is introduced 10 seconds after playback begins in all cases. We evaluate QoE only during the blockage (from T_1 to T_4). Unless otherwise mentioned, we use TCP as the underlying transport protocol. Our evaluation compares PRISM-XR's performance in conjunction the same ABR strategies from the OTA experiments: bandwidth (labeled BW in figures), the buffer-based algorithm (labeled BBA), and LoL$^+$.

Scenario 1: Non-Transient Blocker. The non-transient blocker scenario models either (i) a large blocker which persists beyond the predictor's time horizon or (ii) a mitigation technique that performs an action without further adjustment during the time horizon, e.g., switching the serving AP. This results in a single large drop at the end of the Advance Interval which persists until the end of the EOH Interval, with the Blockage Interval being effectively zero. Fig. 4(a) illustrates the rate during this scenario and labels the intervals.

We evaluate PRISM-XR across rate drops of varying severity by changing the initial throughput. The initial throughput varies from 300-600 Mbps and drops to 50 Mbps 10 seconds after playback begins.

Scenario 2: Transient Blocker. In the transient blocker scenario, the time horizon includes three average throughput levels over three intervals. This scenario models: (i) a small blocker expected to clear within the predictor's time horizon, or (ii) a mitigation technique that takes action before the blockage (e.g., switches the serving AP) and a second action after the blockage clears (e.g., switches back to the original AP). The rate curve for this scenario is shown in Fig. 4(d).

[3]Our parameters were chosen to achieve ultra-low stall times from a depleted buffer state, placing huge emphasis on avoiding rebuffering. Other choices of parameters would reflect different priorities. To better show the impact of our parameter choices, we present individual, unweighted statistics for each of the three terms in our QoE function in addition to the composite score in Section IV-D.

Specifically, the AP-client link starts with a bandwidth of 300 Mbps, drops to 50 Mbps during the blockage 10 seconds after playback begins, then recovers to 300 Mbps. We vary the Blockage Interval from 1-4 seconds.

D. Performance Evaluation Results of PRISM-XR

In this section, we analyze the results of our emulation experiments, beginning with observations from Scenario 1, followed by Scenario 2. Additionally, we discuss the performance of PRISM-XR when operating with QUIC. Finally, we investigate the impact of inaccuracies in blockage predictions.

Impact of Throughput Drop Severity in Scenario 1. n the non-transient blocker scenario, we evaluate the impact of throughput drop severity by setting different initial rates during the test as described above. Fig. 4(b) shows the impact of drop severity on composite QoE, while Fig. 4(c) shows its impact on the visual quality component of the QoE function described earlier. The gain associated with using PRISM-XR is shown stacked on top of the original QoE score. PRISM-XR achieves a 15% composite QoE gain on average, even in the presence of very large drops in rate. Its improvement over ABR strategies without PRISM-XR is largely due to improved visual quality. Predictive intelligence enables PRISM-XR to fill the buffer with high quality segments without running the risk of stalling since it knows when it needs to reduce resource usage.

Impact of Blockage Interval in Scenario 2. To evaluate how well PRISM-XR handles transient blockers, we vary the blockage interval from 1 to 4 seconds. Since the EOH interval time does not change and the visual quality portion of the QoE function is additive, we expect overall QoE to slightly rise over longer Blockage Intervals as more segments can be downloaded during the period of evaluation.

Fig. 4(e) shows the improvement gained by using PRISM-XR in terms of composite QoE stacked on top of the score achieved by using the labeled ABR strategy without PRISM-XR. Fig. 4(f) shows improvements in terms of visual quality. Figs. 4(g) and (h) show switch penalties and stall time side-by-side instead of stacked. PRISM-XR demonstrates gains of approximately 15%–40% in terms of composite QoE, depending on the scenario, with higher gains observed in cases with longer blockage intervals, showing that PRISM-XR can successfully mitigate large changes in the available bandwidth. Further, since we heavily value stall avoidance in our QoE function, PRISM-XR completely avoids stalls in all of our tests, which ABR strategies without PRISM-XR could not achieve in the same environment without significant compromises on quality or switching costs.

PRISM-XR often causes higher quality switch costs than un-augmented ABR. This is to be expected since our QoE parameters tend to value high visual quality and low stall time over smooth switches in quality. In cases where smooth switches in quality are desirable, the coefficient w_2 can be adjusted to force PRISM-XR to plan smoother rate changes.

Impact of Transport Layer (TCP vs. QUIC). We also evaluated PRISM-XR against ABR strategies in Scenario 1 using QUIC. However, we observed that PRISM-XR only occasionally provides gains and may even reduce QoE. For example, Fig. 4(i) shows the results for Scenario 1 over QUIC with and without PRISM-XR at an initial rate of 300 Mbps. We observe that PRISM-XR's composite QoE is similar to the un-augmented BW and LoL+ ABR strategies.

To investigate the cause of PRISM-XR's poor performance, we conducted iPerf measurements to assess raw throughput. We capped the throughput at 500 Mbps, with a drop to 50 Mbps after 10 seconds, using Linux TC, and ran iPerf tests with both TCP and QUIC. Fig. 4(j) shows the throughout results. Under TCP, the rate is measured relatively accurately, while under QUIC, average throughput is only about 200-300 Mbps with significant fluctuations. Because PRISM-XR relies on accurate predictions of average future throughput, large discrepancies introduced by the QUIC protocol can significantly reduce or eliminate PRISM-XR gains.

Impact of Predictor Accuracy. Although most predictors described in existing literature are highly accurate [6], [7], it is also apparent following the experiments using QUIC that predictor inaccuracy can severely degrade PRISM-XR advantage, so we conducted experiments to quantify its impact.

We consider two types of errors. Figs. 4(k) and (l) evaluate the impact of errors in the blockage's start time and the experienced data rate, respectively. The impact is measured as the difference in composite QoE with and without PRISM-XR. To better illustrate sensitivity to these errors, we focus on Scenario 2 under slightly more severe conditions. Specifically, we simulate a 5-second drop occurring 15 seconds after playback begins, where the data rate falls from 300 Mbps to 10 Mbps before returning to 300 Mbps. In all cases, the emulator notifies PRISM-XR 12 seconds after playback starts. The spread of these errors is intended to cover a wide range of possible scenarios, including lateness as a result of e.g. client-side performance bottlenecks such as use of QUIC, or unexpected changes in user behavior.

In the start time error scenario (Fig. 4(k)), we modify the actual blockage start time while keeping the notification unchanged. This type of error may occur when a mobile blocker, such as a bus or a human body, slows down after the notification is sent, causing the blockage to start later than expected while maintaining the same duration. In the rate error scenario (Fig. 4(l)), we modify the data rate the client experiences during the blockage. For instance, instead of dropping to 10 Mbps, the blockage may result in a complete loss of connectivity or be less severe than expected.

The original values—representing a scenario with no error in predicting the blockage start time or rate—are highlighted in red on the x-axis of Figs. 4(k) and (l), corresponding to values 15 second and 10 Mbps, respectively.

In both cases, we find that PRISM-XR provides significant gains (increase in composite QoE value) when no errors are present, as expected. However, these gains can diminish or even reverse in the presence of large errors, with the severity of performance degradation depending on the ABR strategy. For instance, if the expected blockage start time is 15 seconds, the relative QoE gain for BW remains positive even if the actual

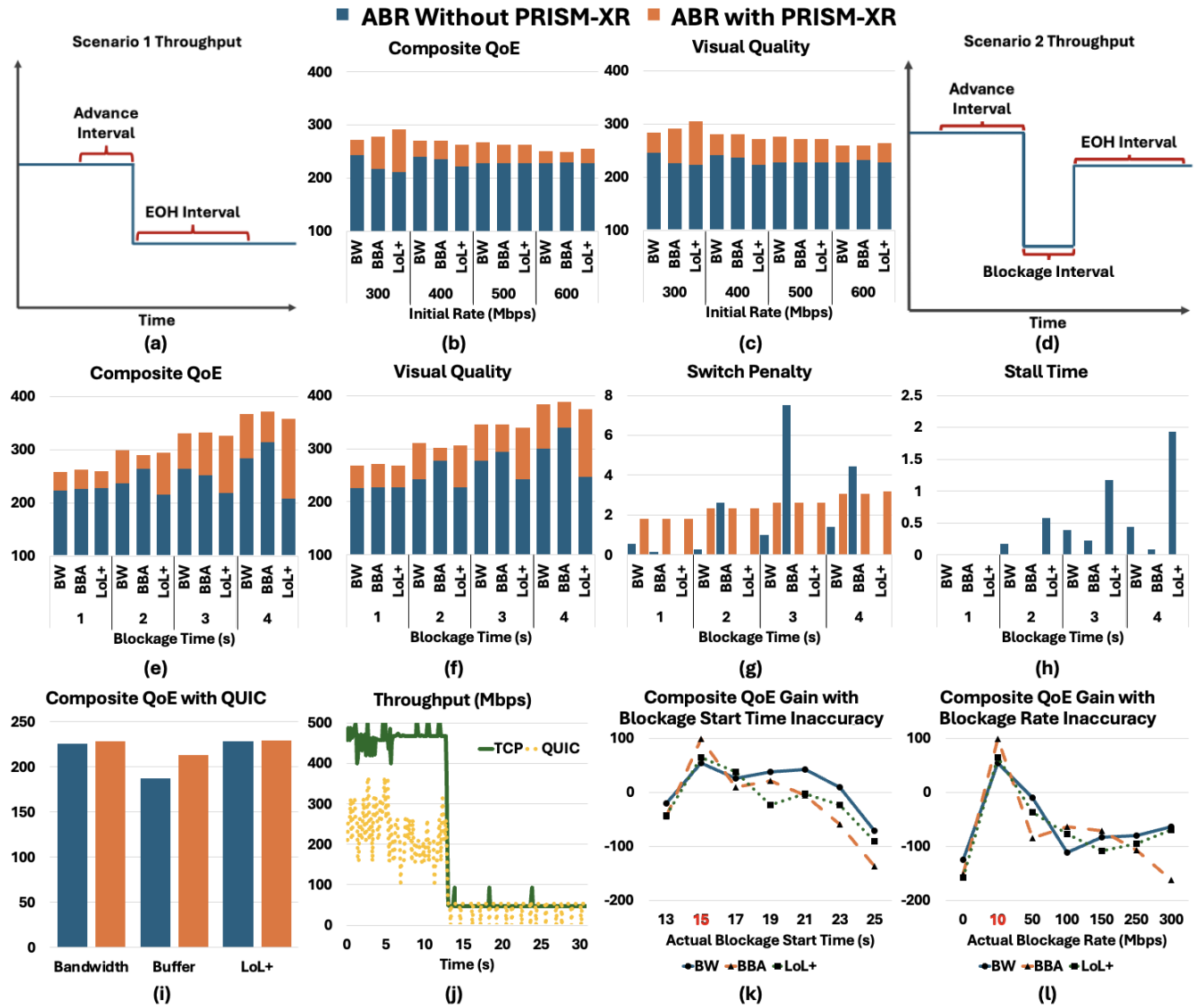

Fig. 4. **The title of each figure indicates the quantity measured on the y-axis. (a): Scenario 1 throughput curve. (b): Composite QoE across Scenario 1 tests. PRISM-XR's improvement is shown stacked on top of the underlying ABR strategy. (c): PRISM-XR's improvement over reference methods in terms of visual quality alone. (d): Scenario 2 throughput curve. (e): Composite QoE across Scenario 2 tests, with PRISM-XR's improvement again shown stacked. (f): Improvement over reference methods in terms of visual quality alone. (g,h): Comparison of switch and stall time penalties, respectively. Results w/wo PRISM-XR are shown side by side for clarity. (i): Composite QoE during blockage when QUIC is used. (j): iPerf throughput traces using TCP (green) and QUIC (yellow). (k,l): PRISM-XR performance when the predictor under- or overestimates the time until the blockage occurs or the blockage rate, respectively. The original values—representing the scenarios with no error—are highlighted in red on the x-axis.**

blockage occurs at 21 seconds (Fig. 4(k)). In contrast, PRISM-XR on top of LoL+ is more sensitive to prediction accuracy, with the relative gain turning negative if the blockage actually occurs at 19 seconds.

Similarly, if the expected blockage rate is 10 Mbps but the actual rate is 50 Mbps (Fig. 4(l)), PRISM-XR's QoE is almost equal to the un-augmented BW QoE. However, BBA drops drastically in presence of a similar inaccuracy.

V. Conclusion and Future Work

MmWave wireless enables new video formats with high visual fidelity and interactivity—both essential for the widespread adoption of Metaverse applications. However, it also introduces challenges that current-generation adaptive bitrate (ABR) strategies are not designed to handle. In this work, we make two key contributions to improving ABR streaming over mmWave links: (i) a physical testbed for analyzing the impact of transport layer protocols on streaming QoE, and (ii) an adaptive bitrate strategy augmentation called PRISM-XR, designed to mitigate the effects of large fluctuations in data rate, which are common in mmWave networks. Our experiments demonstrate that augmenting ABR strategies with physical-layer predictive intelligence can improve video QoE by up to 40% during blockage events.

As part of future work, we plan to take better advantage of features unique to 360° video, such as 360° HMD (Head-Mounted Display)-navigation traces and tiled video encoding as part of our download plan optimization framework.

VI. Acknowledgements

This work was supported in part by the NSF under CNS awards 1942305, 2106150, 2032033, and 2346528.

References

[1] J. Chakareski, M. Khan, and M. Yuksel. Towards enabling next generation societal virtual reality applications for virtual human teleportation. *IEEE Signal Processing Magazine*, 2022.

[2] Y. Kumar and T. Ohtsuki. Influence and mitigation of pedestrian blockage at mmwave cellular networks. *IEEE Transactions on Vehicular Technology*, 2020.

[3] A. Almutairi, A. Keshavarz-Haddad, and E. Aryafar. A deep learning framework for blockage mitigation and duration prediction in mmwave wireless networks. *Elsevier Ad Hoc Networks Journal*, 2024.

[4] S . Wu, M. Alrabeiah, C. Chakrabarti, and A. Alkhateeb. Blockage prediction using wireless signatures: Deep learning enables real-world demonstration. *IEEE Open Journal Communications Society*, 2022.

[5] M. Alrabeiah and A. Alkhateeb. Deep learning for mmwave beam and blockage prediction using sub-6 ghz channels. *IEEE Transactions on Communications*, 2020.

[6] S. Wu, C. Chakrabarti, and A. Alkhateeb. Proactively predicting dynamic 6G link blockages using LiDAR and in-band signatures. In *IEEE Open Journal Communications Society*, 2023.

[7] J. Zhang, H. Xie, and H. Wang. User-centric phaseless beam and blockage prediction empowering 5G mmWave systems. In *IEEE Transactions on Communications*, 2024.

[8] 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 *Proceedings of ACM SIGCOMM*, 2014.

[9] 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.

[10] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli. A control-theoretic approach for dynamic adaptive video streaming over HTTP. In *Proceedings of ACM SIGCOMM*, 2015.

[11] 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 Transactions on Multimedia*, 2021.

[12] H. Mao, R. Netravali, and M. Alizadeh. Neural adaptive video streaming with pensieve. In *Proceedings of ACM SIGCOMM*, 2017.

[13] 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 *Proceedings of the 1st Workshop on 5G Measurements, Modeling, and Use Cases (5g-MeMU)*, 2021.

[14] S. Shippey, S. Srinivasan, E. Aryafar, and J. Chakareski. An experimental evaluation of 360-degree ABR video streaming over mmwave wireless links. In *Proceedings of IEEE MetaCom*, 2024.

[15] X. Zhang, S. Jin, Y. He, A. Hassan, Z.M. Mao, F. Qian, and Z.L. Zhang. QUIC is not quick enough over fast internet. In *Proceedings of ACM WWW*, 2024.

[16] B. Jaeger, Johannes Zirngibl, Marcel Kempf, Kevin Ploch, and Georg Carle. QUIC on the Highway: Evaluating Performance on High-rate Links. In *Proceedings of IFIP Networking Conference*, 2023.

[17] A. Almutairi, S. Srinivasan, A. Keshavarz-Haddad, and E. Aryafar. Deep transfer learning for cross-device channel classification in mmwave wireless. In *Proceedings of IEEE MSN*, 2021.

[18] A. Ansari and M. Wang. iStream Player: A versatile video player framework. In *Proceedings of the 33rd NOSSDAV Workshop*, 2023.

[19] B. Lantz, B. Heller, and N. McKeown. A network in a laptop: Rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets)*, 2010.

[20] quic-go. https://github.com/quic-go/quic-go.

[21] X. Liu, Y. Huang, L. Song, R. Xie, and X. Yang. The SJTU UHD 360-Degree Immersive Video Sequence Dataset. In *Proceedings of IEEE ICVRV*, 2017.

[22] Quictun. https://github.com/gnolizuh/quictun.