

RESEARCH ARTICLE

Edge-Computing-Assisted Virtual Reality Computation Offloading: An Empirical Study

BARAKA WILLIAM NYAMTIGA¹, AIRLANGGA ADI HERMAWAN¹, YAKUB FAHIM LUCKYARNO¹, TAE-WOOK KIM², DEOK-YOUNG JUNG², JIN SAM KWAK³, AND JI-HOON YUN¹

¹Department of Electrical and Information Engineering, Seoul National University of Science and Technology, Seoul 01811, South Korea

²Clicked Inc., Seoul 03920, South Korea

³WILUS Institute of Standards and Technology, Seongnam, Gyeonggi 13595, South Korea

Corresponding author: Ji-Hoon Yun (jhyun@seoultech.ac.kr)


This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP), South Korea, funded by the Korean Government [Ministry of Science and ICT (MSIT)] [Biosignal-Based Predictive Content Creation for Cloud Virtual Reality (VR)] under Grant 2017-0-00650.

ABSTRACT Offloading heavy virtual reality (VR) computational operations to a network edge computation entity is receiving increasing attention as a tool to wirelessly and energy efficiently provide low-end client devices with high-quality and immersive interactive VR services anytime and anywhere across the globe. In this work, we aim to provide an understanding of various characteristics of VR computation offloading through comprehensive experiments conducted using a prototype testbed for edge-assisted VR processing and streaming. First, we investigate the benefits of VR offloading in terms of computational load and power consumption reduction for a client device compared to standalone operation. Next, we measure VR traffic patterns, including frame size and data and packet rates with various settings, such as different resolution and encoding options. We also measure several performance metrics associated with the quality of experience, namely, frame rate, packet loss rate, and image quality, with various configuration settings. Then, we present latency measurement studies and investigate per-component latency with various settings. Furthermore, we report the rigorous experiments performed to study the impacts of latency and motion patterns on the black borders formed due to image reprojection and the overfilling technique used to eliminate these black borders.

INDEX TERMS Virtual reality, edge computing, offloading, VR streaming, latency, overfilling.

I. INTRODUCTION

Virtual reality (VR), a three-dimensional (3D) environment allowing a user to enter and interact with alternate realities [1] that are rendered through the utilization of audio-visual components coupled with other sensory devices [2], can exist in various forms. Such forms include interactive environments, omnidirectional video, or a hybrid of the two forms combining computer-generated scenes with natural scenes. An interactive environment is a form of VR where data models and algorithms are utilized to generate a synthetic scene in real-time in accordance with the user's head pose and other input triggers. Omnidirectional video, however, is a form of VR wherein a special camera and several microphone devices

The associate editor coordinating the review of this manuscript and approving it for publication was Lei Shu .

are used to capture a natural scene to produce a panoramic video resembling that presented by a conventional TV system but having an unrestricted viewing arc.

To realize an interactive VR environment offering complete user immersion, a high-quality visual experience and seamless navigation in the virtual world are required to mimic the user's experience of the real world [3]. These requirements are fulfilled using dedicated graphics cards equipped with abundant GPU cores to render the graphics. Although VR technology is being rapidly adopted in the consumer domain, a number of obstacles still hinder its massive-scale overall adoption. The high costs of the computing and graphics processing hardware, low visual quality and constrained user mobility are among the notable bottlenecks [4]. The conventional usage of *tethered* VR, where a wire harness or a direct wireless link (e.g., VIVE wireless adapter [5])

connects the VR headset to a local PC, makes the approach cumbersome to set up, is relatively expensive and is not cost effective for single-use scenarios. The costs arise from the high price of VR headsets (both PC-based and console-based) in addition to the expensive high-end PC needed to play back the interactive content. Consequently, the audience is limited to only those in ownership of suitable platforms. Moreover, the wire harnessing a tethered VR headset to the processing PC restricts the user's freedom of movement, and thus, impacts usability. In contrast, *standalone (untethered)* VR headsets make use of built-in processing units, and thus, offer greater convenience and portability of VR services. However, they require additional local processing capability and battery power to render VR content of high quality; otherwise, only limited-quality VR content is available on these devices.

As a result of this, an approach for offloading all or most VR computational operations to one or more remote GPUs of a network edge computation entity, as depicted in Fig. 1, which we call *edge VR* in this paper, is receiving increasing attention.¹ Offloading heavy VR computations to an edge computing entity possessing sufficient computational power reduces the need for local deployment of a powerful PC. This offers a promising approach for overcoming the highlighted obstacles hindering the large-scale adoption of VR by a larger audience wirelessly connected to the internet and allowing them to experience immersive services from their desired devices anytime and anywhere across the globe [4], [6]. The abundant processing and storage resources found at the edge can make high-quality and pay-as-you-go VR services available to a large group of online VR users. With the edge providing the necessary resources to execute and render scenes, the client headsets are left with only the task of displaying the content streamed over wireless networks. This makes it possible to manufacture lighter and cheaper thin client VR headsets that are more affordable and sustainable. Higher user mobility is facilitated, and with only light operations performed by the headsets, their battery life is also expected to be extended.

However, the offloading of VR computations to the edge is susceptible to *latency*, which presents a major challenge [7], mainly due to the transmission of VR traffic over bandwidth-limited networks. Although new wireless technologies, such as fifth-generation (5G) New Radio (NR) [8] and IEEE 802.11be [9] aim to increase bandwidth and reduce latency, the large volume of raw VR display data resulting from its ultrahigh resolution is still not fully compatible with the network bandwidths of present and near-future wireless network and internet systems. This situation inevitably necessitates the encoding (compression) and decoding (decompression)

¹The difference between wirelessly tethered VR and edge VR is the capability of a networked connection between a VR headset and a computing entity. Through a networked connection, edge VR is immediately applicable to various network scenarios, allowing a VR headset to connect to the edge anywhere anytime under Wi-Fi or cellular coverage and the edge located anywhere in the network. In contrast, wirelessly tethered VR only provides a direct connection between a VR headset and a local PC on a room scale and is not immediately extensible to network scenarios.

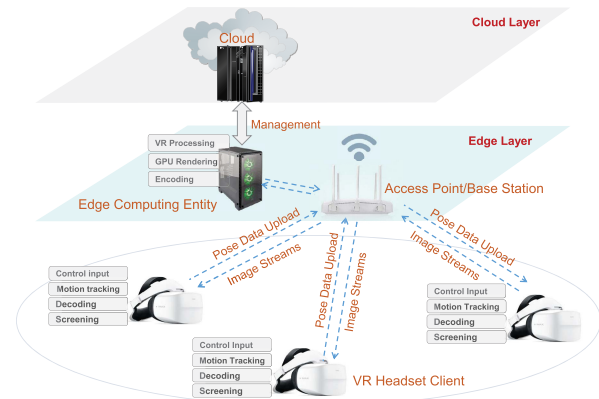


FIGURE 1. Edge VR layered architecture.

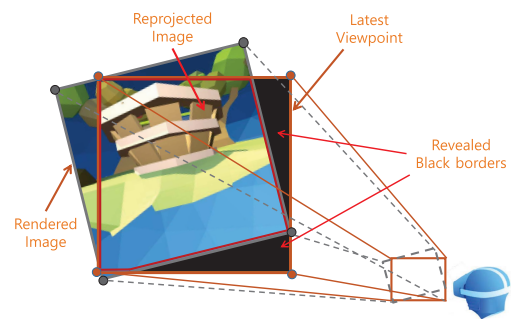


FIGURE 2. VR image reprojection and resulting black borders.

of the raw VR display data on the edge and client sides, respectively, incurring more latency in addition to the network transport latency. As the latency increases, the inconsistency between the rendered VR image and the user's viewpoint at the time of scan-out also increases. A large mismatch leads to unpleasant physiological symptoms due to the lag between the sensory inputs and the visual and vestibular systems, commonly described as *motion sickness*, and is likely to cause the user to ultimately quit using the service. Up to 20 ms of *motion-to-photon latency* is widely considered acceptable for VR applications, as this level of mismatch is undetectable by the user, and thus, does not cause motion sickness [7], [10]. Moreover, other forms of discomfort, such as unsatisfactory image clarity, image smearing, and dizziness, can also be experienced in high-latency VR environments [11]. The *image reprojection* technique (also called time warping) [12], [13], [14] is essential for minimizing the motion-to-photon latency to remedy the motion sickness problem in VR. As illustrated in Fig. 2, in this technique, a rendered frame is reprojected before being scanned out to reflect the movement in the head pose that occurs after the scene is rendered [15]. With image reprojection, however, if the end-to-end latency level is excessive, a large number of black borders will be produced after reprojection, disrupting user immersion [2].

To achieve smooth and comfortable immersion in VR, good platform designs are vital for meeting a range of application requirements at acceptable levels of latency [16]. This first requires a deep quantitative understanding of the characteristics of VR offloading systems. However, there has been no in-depth empirical study of edge VR in the literature. In addition, the expected benefits of edge VR also need to be proven and measured through empirical studies. There have been a few testbed studies of VR offloading and streaming [7], [17], but they only provided brief results focusing on overall performance. Moreover, no study has demonstrated the impact of latency on perceived quality in VR. Empirical studies on cloud gaming [18], [19] have not considered VR applications, and thus, do not provide VR-specific insights. Therefore, comprehensive empirical studies spanning a range of VR-specific application requirements for interactive VR environments are needed to gain an understanding and support the design of edge VR systems.

In this work, we aim to provide an understanding of VR computation offloading through comprehensive testbed experiments addressing various aspects of the situation. For the experiments, we prototype a testbed of a VR computation offloading system using existing technologies and use it to conduct measurement studies. First, we investigate the benefits of VR offloading in terms of computational load and power consumption reduction for a client device compared to standalone operation. Next, we measure VR traffic patterns, including frame size and data and packet rates under various settings, such as various resolution and encoding options. We also measure several performance metrics associated with the quality of experience, namely, frame rate, packet loss rate, and image quality under various configuration settings. Then, we conduct latency measurement studies and investigate the per-component latency under various settings. Furthermore, we perform rigorous experiments to study the impacts of latency and motion patterns on the black borders formed due to image reprojection and the overfilling technique used to eliminate these black borders.

The main contributions of our work and the insights obtained from the experimental results are summarized as follows:

- We prototype a testbed of a VR computation offloading system using the image reprojection and overfilling techniques to combat latency and black borders and to conduct experimental measurements based on real VR user motion data.
- We investigate the benefits of VR offloading in terms of computational load and power consumption reduction compared to standalone operation, which shows that VR offloading can reduce computational load by up to 74% and power consumption by up to 27%.
- We measure VR traffic patterns, quality metrics, and latency. We demonstrate that both the VR traffic patterns and quality are strongly affected by the encoder configuration. We also show that the encoding and decod-

ing latencies predominate, while the network transport latency is also not negligible.

- We study the impacts of latency and motion patterns on the formation of black borders. Next, we show that angular changes increase with increasing latency, leading to increasing black borders, and that overfilling successfully reduces black borders at the cost of increasing computational overhead with increasing latency.

The rest of this paper is organized as follows. Recent studies related to edge-computing-assisted VR services are reviewed and discussed in Section II. The testbed system used for experiments and measurements is described in Section III. Section IV reports the VR traffic pattern results. Sections VI and VII present the quality and latency results, respectively. Experimental results illustrating the influence of motion patterns, the phenomenon of black borders, and the effect of overfilling are provided in Section VIII, and the conclusion is given in Section IX.

II. RELATED WORKS

A. CLOUD COMPUTING FOR GAMING AND VR

Cloud and edge computing has been extensively studied for a wide range of tasks, such as big data processing [20], task offloading for the Internet of Things (IoT) combined with blockchain technology [21], [22], and offloading of general tasks while leveraging between throughput and fairness [23]. Another use case is gaming for providing high-quality services with lightweight user-side computing [24]. Sharing computing resources, such as CPU and GPU resources between virtual machines for cloud gaming was discussed in [25] and [26]. In addition, it has been demonstrated that a cloud gaming platform can provide a 3D gaming experience at an acceptable latency [27]. Interactive remote rendering systems that have been proposed in the literature were surveyed in [28].

There have been several proposals aimed at reducing latency in cloud gaming and VR. Outatime [29] predicts future user actions and renders multiple frames for probable actions in the near future based on the user's historical and recent behavior. The side effects of prediction failures are prevented by a process checkpoint and rollback service. Furion [30], Kahawai [31] and Cloud Baking [32] distribute the rendering loads between the server and the client to reduce latency and save network bandwidth for streaming. FlashBack [33] relies on an installed cache to store a set of costly prerendered frames identified by camera poses. If a user encounters a scene matching one of the cached frames, the scene in the nearest-placed cache is retrieved to avoid the rendering cost for that frame.

B. VR OFFLOADING

Offloading VR processing to a remote host and bandwidth-efficient streaming of VR contents to user devices have been studied mostly for 360-degree VR video [34]. To reduce the heavy bandwidth consumption resulting from the provision

of full-degree information, each picture frame of a 360 video can be spatially split into rectangular regions, called *tiles* in H.265 [35], and only a minimal subset of the tiles covering the user's current viewpoint can be transferred by a VR video server [36], [37], [38], [39], [40]. Simultaneously, transferring a low-resolution layer of the video for a wider area or for the entire area can also be considered [41], [42]. Moreover, a heuristic method of probabilistic viewpoint prediction using curve fitting was proposed in [43]. In [44], the probability distribution of the fixation-point prediction error was derived as a normal distribution under certain assumptions, and the region where the future viewpoint was likely to exist at a given confidence level was obtained in a closed form. Feng *et al.* [45] proposed adaptive user preference modeling and word embedding to dynamically select the video viewpoint at runtime based on the user's head orientation. Mehrabi *et al.* [46] proposed utilizing multiaccess edge computing to jointly optimize the tradeoff between the average video quality and delivery latency by controlling the amount and quality of the streamed content.

Enhancing streaming protocols is another way to reduce the latency and bandwidth consumption of cloud VR. Adaptive video streaming [47] is an application-layer streaming protocol for cloud gaming that exploits selective frame transmission (dropping low-weight frames) and forward error correction for high-weight frames (I and P frames). Shi *et al.* [48] designed a method of selecting key frames to be encoded at a high bit rate, while others are encoded at a low bit rate, thereby reducing resource consumption. A multipath Transmission Control Protocol (TCP)-based streaming framework for 360-degree VR videos that dynamically selected the appropriate tile bit rate in accordance with the bandwidths and transmission delays of different subflows was proposed in [49]. Chen *et al.* [50] proposed a streaming framework for 3D assets in VR services based on user gaze behaviors.

Some research works solved radio resource management problems for the offloading of VR services [51], [52], [53], [54]. Chen *et al.* [55] solved the resource management problem for wireless VR in cellular networks by exploiting the potential spatial data correlations among users due to their engagement in the same VR environment to reduce the traffic loads in both the uplink and downlink directions. Guo *et al.* [54] solved a similar problem using distributed learning in millimeter wave (mmWave)-enabled wireless networks with mobile edge computing. Dang and Peng [52] solved a joint radio communication, caching and computing decision problem to maximize the average delay tolerance at both mobile VR devices and fog access points. Huang and Zhang [56] proposed a multiuser medium access control (MAC) scheduling scheme with a low-complexity downlink user selection algorithm for VR services in a 5G system.

C. WIRELESS VR

There have been recent attempts to design communication and networking schemes for wireless VR in unlicensed spec-

tra. Abari *et al.* [57] proposed MoVR to solve the signal blocking problem in the 60-GHz band by reflecting signals toward the user. Kim *et al.* [58] proposed a dynamic and adaptive algorithm that could control the power allocation in 60-GHz transceivers to achieve time-averaged energy efficiency for VR data delivery while preserving queue stability. In [59], the feasibility of wireless VR using WiGig was examined through performance measurements and simulation studies. In [60], the feasibility of wireless VR over Wi-Fi was examined via testbed experiments, and the challenges were discussed. Ahn *et al.* [61] proposed securing timely transmission opportunities by using trigger-based transmission. Tan *et al.* [62] proposed several enhancement schemes for the Wi-Fi MAC protocol to better support motion feedback for wireless VR, including prioritizing older motion data, obtaining motion feedback using the reverse direction, and limiting the aggregation size. Kim *et al.* [63] proposed a motion-aware interplay mechanism for WiGig and Wi-Fi to achieve higher perceived quality and connection reliability. Considerations regarding the delivery of VR services over cellular networks, such as 5G systems have also been addressed. Elbamby *et al.* [16] discussed the challenges and enablers for ultrareliable low-latency wireless VR, including edge computing and proactive caching in mmWave cellular networks.

D. VR STREAMING TESTBED

There have been a few testbed studies of VR offloading and streaming. In [7], a VR streaming platform was prototyped, and performance measurements were made. The authors suggested optimizations for higher frame rates and lower bandwidth consumption. In addition, they proposed a dynamic transfer of small objects to the client device at runtime to provide shorter interaction latency. Rohloff *et al.* [17] implemented an open-source framework with a customized network stack to eliminate unnecessary memory operations incurred by mismatching data formats in each layer. Furthermore, Xiao *et al.* [64] implemented a VR streaming framework using Unreal Engine 4 and applied fixed-foveated rendering technology. Through experimental results, they showed that it reduces overall latency and increases the frame rate.

However, these studies provided only brief and general performance results. Moreover, no study has shown the impact of latency on perceived quality in VR and the effect of the overfilling technique for varying latency. Therefore, the salient point of our work is an in-depth investigation of the characteristics of VR offloading in various aspects, for which a testbed of a VR computation offloading system using the image reprojection and overfilling techniques is prototyped and the characteristics and performance benefits are measured.

III. VR OFFLOADING SYSTEM TESTBED

The architectural components of our testbed system are displayed in Fig. 3. The upper part of the figure shows the

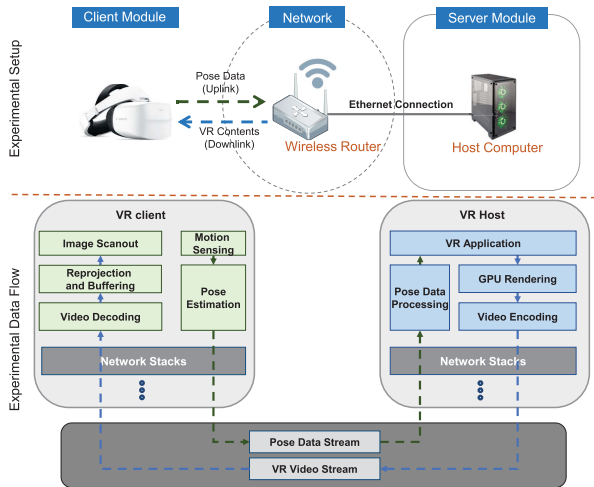


FIGURE 3. Experimental VR offloading testbed system.

general experimental setup with all pieces of the hardware equipment used and their interconnections, whereas the bottom part depicts the data flow across the different components of the system. Unless specified otherwise, the host computer is equipped with an Intel Core i9-10900K CPU @ 3.7 GHz (10 cores), 32 GB of RAM running on a solid-state drive (SSD), Windows 10 and an NVIDIA graphics card (RTX 3080) that is capable of hardware acceleration for video encoding. The rendering engine and encoder are set to generate 60 frames per second, i.e., one frame every 16.7 ms. The host computer is connected to a wireless router using a Gigabit Ethernet connection. The client VR headset is the Oculus Quest and is connected to a wireless router via an IEEE 802.11ac Wi-Fi interface. It is located close to the router so that the highest link speed of the Wi-Fi interface is always used. The headset periodically sends the user’s latest pose information to the host computer at 60 Hz. The streaming protocol of VR frames (including audio data) is a custom-built protocol using the User Datagram Protocol (UDP) and that of the pose information is the TCP-based ZeroMQ messaging protocol [65].

As illustrated in Fig. 4, the operational pipeline of the system begins with the user’s motion being sensed using built-in motion tracking sensors in the VR headset. The sensor data samples are forwarded to the pose estimation engine to determine the user’s current pose [66]. The pose data are then transmitted to the application server, based on which the server performs 3D simulation and renders a VR viewpoint image corresponding to the pose. The image output by the renderer is fed into a video encoder to be compressed prior to being transmitted back to the client headset. At the client headset, frame packets are received and assembled into frames. The frame data are then passed to the video decoder of the headset. After being decoded, the current frame image is read from the decoder’s output buffer and reprojected based on the latest pose information retrieved from the pose estimation engine. The reprojected image is then placed in the

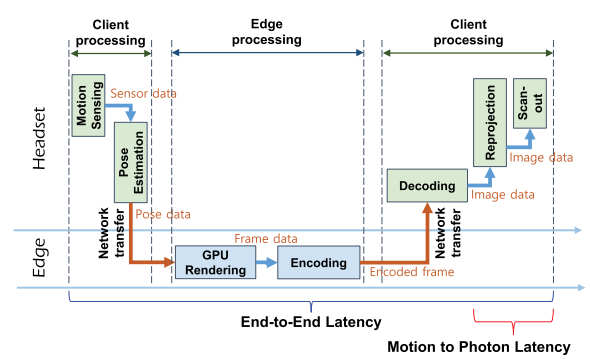


FIGURE 4. Operational pipeline of the VR streaming testbed system.

TABLE 1. Experimental setup.

Item	Configuration
Host hardware specification	i9-10900K, 32 GB RAM, RTX 3080 (PC2: i7-9750H, 32 GB RAM, RTX 2060)
VR image resolution	[1, 2, 4]K pixels
Encoding codec	[H.264, H.265]
Bit rate of encoding	[1, 30, 100] Mbps
Quantization parameter	[18, 28, 35]
VR headset	Oculus Quest 1 (and Quest 2)
Headset connection	IEEE 802.11ac
Frame rate	60 Hz
Head pose reporting rate	60 Hz
Induced latency	[50, 100, 200, 300] ms
Overfilling factor	[1, 1.2, 1.4, 1.6]

frame buffer and is finally scanned out via the headset display at the native refresh rate of the display.

IV. OFFLOADING EFFECTS IN TERMS OF COMPUTATIONAL LOAD AND POWER CONSUMPTION

We present comparisons of the computational loads and power consumption for two different operating modes of the client when playing VR games, namely, the standalone and offloading modes, which are also compared against the idle state (no content being played). To evaluate the computational and power performance of the two modes on the same computing platform, a x86 single-board computer (an Intel Z8350 Quad Core CPU operating at up to 1.92 GHz with 4 GB of RAM) is used. Three different games (Armagetron Advanced [67], Great Power [68] and Roller Coaster [69]) are used for experimentation. During gameplay, the CPU and power metrics are logged by the HWINFO diagnostic software tool [70] and a Wattman power meter [71], respectively.

The experimental results in Fig. 5, presented in the form of the cumulative distribution function (CDF) curves and the corresponding average bar charts in Fig. 6 both show that the offloading mode substantially reduces both CPU and power resource consumption compared to the standalone mode. Regardless of the played contents, the levels of dissipation of CPU and power resources are lower in the offloading mode. Fig. 6 shows that the offloading mode consumes from 23%

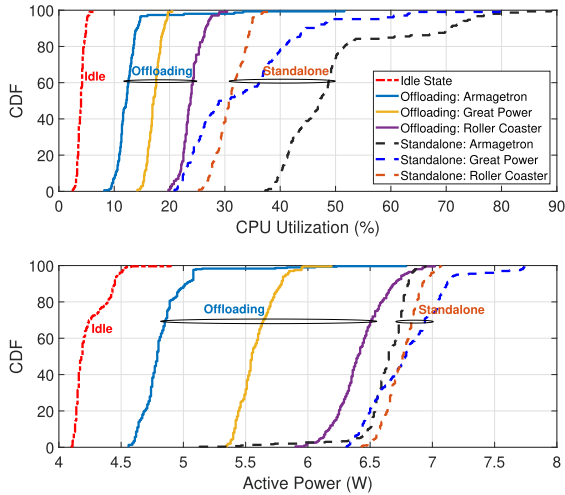


FIGURE 5. CDFs of CPU load (top) and power consumption (bottom) for a client device operating in standalone vs. offloading mode.

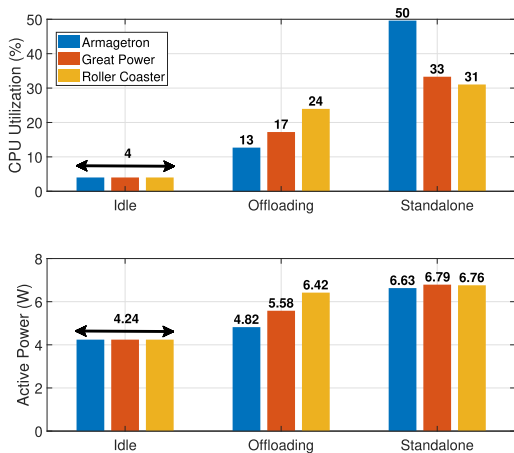


FIGURE 6. Mean values of CPU utilization (top) and power consumption (bottom) for different games in different operating modes.

(Roller Coaster) to 74% (Armagetron) less CPU power and from 5% (Roller Coaster) to 27% (Armagetron) less battery power than the standalone mode for playing the same content. These gains suggest that the offloading mode is a promising tool for solving the CPU and the battery power problems of VR services. Notably, the CPU loads in the standalone mode show marked variations among the different games, as seen in Fig. 5, whereas the power consumption in the standalone mode is similar for all three games.

V. VR STREAMING TRAFFIC PATTERNS

We present observations of the characteristics of VR streaming traffic from the server to the client, including packet rates, bit rates and frame sizes. To evaluate traffic patterns in VR streaming, we play the Showdown VR demo [72]²

²Results gathered from the Showdown VR demo [72] and from the Acan’s Call VR game [73] demonstrate quite similar patterns. Thus, only the results for the Showdown VR demo are presented.

and observe the resulting patterns in the bandwidth being consumed, the size of the video frames and the packet rate during streaming. We conduct experiments under different configurations in terms of the bit rate, the encoding standard and the video resolution with which compression is applied prior to streaming the content. The H.264 advanced video coding and H.265 high efficiency video coding standards are used for video compression, while 1, 30, and 100 Mbps are selected as representative encoding bit rates. We consider 2K (2560 × 1440) and 4K (3840 × 2160) resolutions. The primary goal of video encoding is to save as many bits as possible relative to the original data while maintaining acceptable quality. The method used for rate control plays a vital role in establishing the tradeoff between the payload size and the quality of the resulting video. Among the many rate control methods available, we consider the constant bit rate (CBR) mode and the constant quantization parameter (QP) mode as one possible form of a variable bit rate mode in our experiments. The metrics we obtain from the experiments to characterize VR traffic patterns are listed below:

- *Encoded frame size*: This is the data size of each VR frame after encoding, which is expressed as the number of bytes.
- *Bandwidth usage (data rate)*: This is the number of data bytes streamed over the VR service connection per unit time. It corresponds to the network bandwidth required to transmit the VR video stream.
- *Packet rate*: This is the number of packets generated for the VR video stream per unit time.

Fig. 7 displays the time evolution and CDFs of the encoded frame sizes for the different codecs and for different bit rates in the CBR mode. From these curves, we can see that the applied bit rate for CBR encoding considerably affects the encoded frame size, but the sizes of the produced frames also show strong fluctuations over time. The fluctuation in the frame sizes becomes more severe at 4K resolution. The CDFs also show that the distribution of the frame sizes is more dispersed for a higher bit rate and a higher resolution. Moreover, the results indicate that the frames generated by the H.264 encoder are slightly larger than those encoded with H.265, a trend that is observed for both the 2K and 4K resolutions. The bar charts in Fig. 8 illustrate a summary of the average sizes of the frames generated under different configurations. The findings demonstrate that the CBR mode is effective in the average sense and that the resolution only minimally affects the average frame size under a CBR configuration. While H.264 tends to generate larger frames than H.265, the difference is quite small. We observe that when the constant QP mode is used for rate control, as seen in Fig. 9, lower QP values result in larger frames, while the fluctuation trends are similar for different QP values.

The experimental results therefore indicate that when more bits are used for encoding, larger frames are produced, and the graphics quality is consequently better at the expense of more bandwidth being consumed. However, large QP values imply

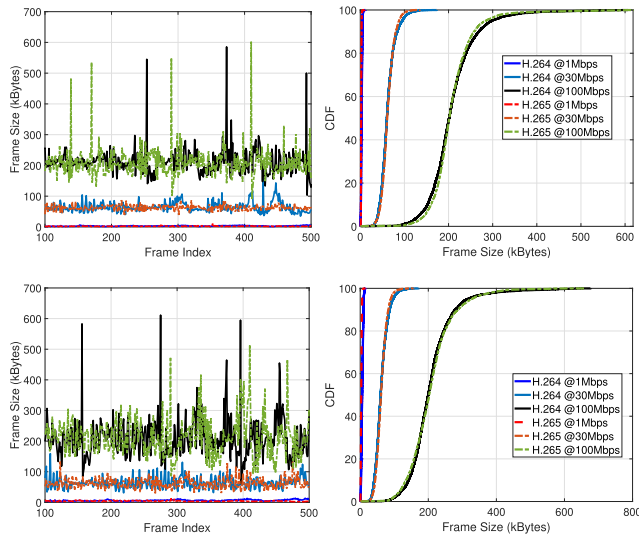


FIGURE 7. Frame size evolution and CDF curves for comparison of compression at 2K (top) and 4K (bottom) resolutions under different encoder and bit rate configurations.

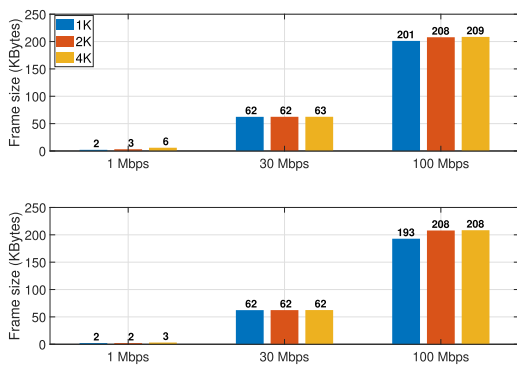


FIGURE 8. Average frame size comparison between the H.264 (top) and H.265 (bottom) encoders under different encoding bit rates and resolution settings.

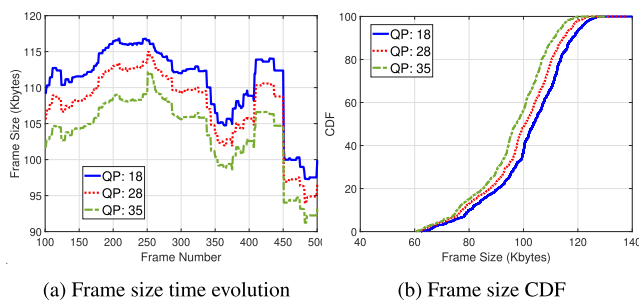


FIGURE 9. Comparison of frame sizes when the constant QP mode is used to control the data rate and H.265 is used for encoding.

that more compression is applied, which leads to smaller frames and quality degradation. Moreover, it can also be seen that the H.265 encoder produces smaller frames than its H.264 counterpart, making it a better choice for efficient bandwidth utilization.

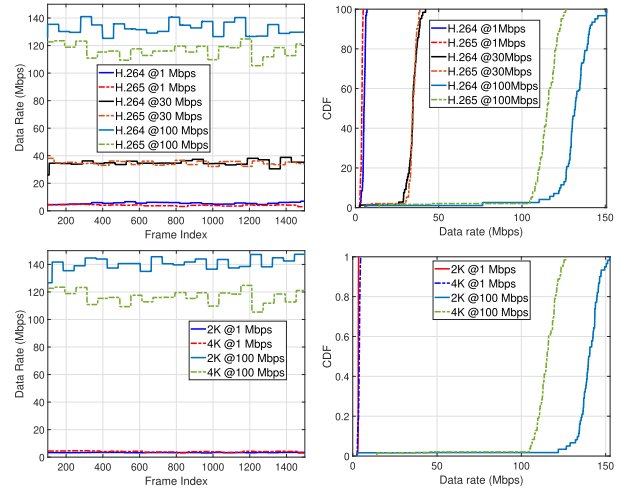


FIGURE 10. Data rate comparisons when 4K VR videos are streamed after compression using different configured bit rates and encoding standards (top) and when the common codec H.265 is used to encode 2K and 4K videos at 1 and 100 Mbps bit rates (bottom).

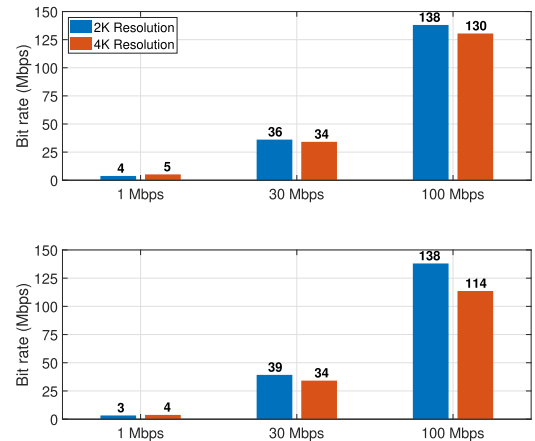


FIGURE 11. Average data rates when streaming VR contents at various resolutions under different bit rates using H.264 (top) or H.265 (bottom) for content compression.

Fig. 10 depicts the time evolution and the CDF curves of the data rates when the VR streams are transmitted after being encoded using various settings in terms of bit rate, resolution and encoding standard. The average results are given in Fig. 11. H.265 shows slightly lower bandwidth usage than H.264 at 1 and 30 Mbps. However, the gap increases substantially at 100 Mbps and 4K resolution, which is the result of a reduced frame rate, as is shown in the next section. When 4K videos are encoded at 1 Mbps, they tend to consume more bandwidth than videos at 2K resolution.

Fig. 12 reveals the general trend of the rate at which the packets are transmitted from the server to the VR device during streaming. It can be observed that the packet rate of H.264 increases with an increase in resolution at 1 Mbps even though the configured rate is fixed at 1 Mbps. In comparison, H.265 shows a much lower increase in the packet rate with an increasing resolution. Meanwhile, the trends at 100 Mbps

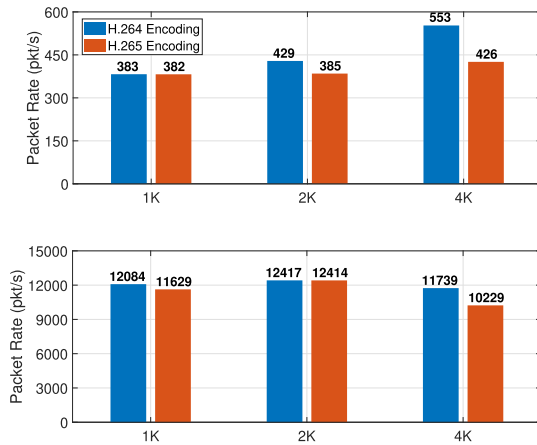


FIGURE 12. Packet rate statistics under different configurations in terms of resolution and encoding standard when configured bit rates of 1 Mbps (top) and 100 Mbps (bottom) are used.

are different. Both codecs show decreasing packet rates with increasing resolution. This is again attributed to a decreasing frame rate. H.265 shows a larger decrease than H.264 because it has a larger decrease in its frame rate.

VI. VR STREAMING QUALITY OF EXPERIENCE

The quality of experience for VR services is also affected by the quality of the graphics displayed to the user [18], [74]. To quantify the graphical quality of VR streaming, we consider the following metrics:

- *Frame rate:* This is the number of VR frames received by the user device over the VR video stream, measured in frames per second (FPS). To measure the frame rate, the user device counts the total number of received VR frames during a play and divides it by the play time as follows:

$$R_{\text{frame}} = \frac{N_{\text{client-received-frames}}}{\Delta t_{\text{playtime}}} \quad (1)$$

VR content delivered at a sufficiently high frame rate is able to convince the user’s brain that the user is fully immersed in the virtual world [75], [76]. A typical minimum target frame rate is 60 FPS, while 90 or 120 FPS is preferred for higher immersion. We set the frame generation rate of the VR server to 60 FPS in our experiments.

- *Packet loss rate:* Packet losses result in missing data in the corresponding VR frame. Large packet losses may produce visual artifacts in the decoded frames. If packet losses become excessive, decoding failure of the corresponding or consecutive VR frame(s) is also possible.
- *Image quality:* We use the peak signal-to-noise ratio (PSNR) and the structural similarity index (SSIM) as metrics for evaluation. When conducting these experiments, we compare a minute-long segment of the original VR video content prior to compression with a compressed video of the same length as each is being

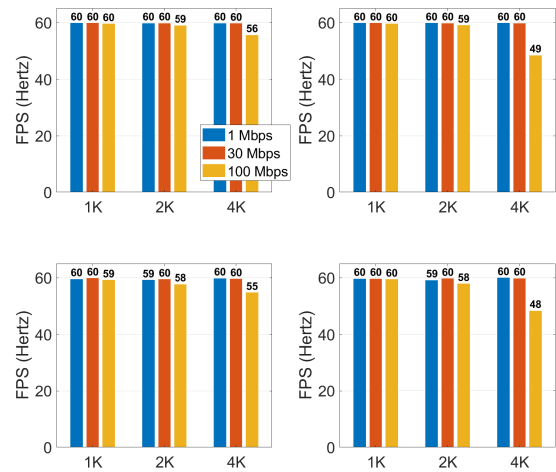


FIGURE 13. Server (top) and client (bottom) frame rate statistics when H.264 (left) and H.265 (right) are used at different resolutions and bit rates for encoding.

streamed to the client under different settings. The metrics are measured using FFMpeg [77].

A. FRAME RATE

Fig. 13 presents the frame rate statistics generated by the server and is received by the client during streaming under different settings in terms of resolution, encoding standard and bit rate of encoding. It is clear from this figure that when compression is performed using a bit rate of 1 Mbps, a frame rate of 60 FPS is consistently achieved for both encoder types at all resolutions and at both the client and server ends. For 2K and 4K resolutions, bit rate settings of 1 and 30 Mbps result in frame rates of 60 FPS at the server and 59-60 FPS at the client. At a bit rate of 100 Mbps, a frame rate of 59 FPS is achieved at the server, but a slightly lower frame rate of 58 FPS is observed on the client side. However, for 4K resolution at 100 Mbps, the frame rate at the server drops to 56 FPS for H.264 and 49 FPS for H.265. This is because the processing overhead of encoding becomes excessive; thus, the server fails to generate frames at the target frame rate. In this case, H.265 has a lower frame rate than H.264 due to its higher processing overhead and possible suboptimality of the codec implementation. The client receives the data at a frame rate that is 1 FPS lower than the generated frame rate due to packet losses.

B. PACKET LOSS RATE

Fig. 14 shows the packet loss rates measured at the client. As we use a higher bit rate and a higher resolution, the packet loss rate generally increases. The packet loss rate is highest at a bit rate of 100 Mbps compared to the other cases. It is noted that packet losses mostly occur in the network buffers when the number of packets exceeds the network buffer size. At 100 Mbps, excessive data packets are generated, and thus, some are dropped due to buffer overflow. In particular, the

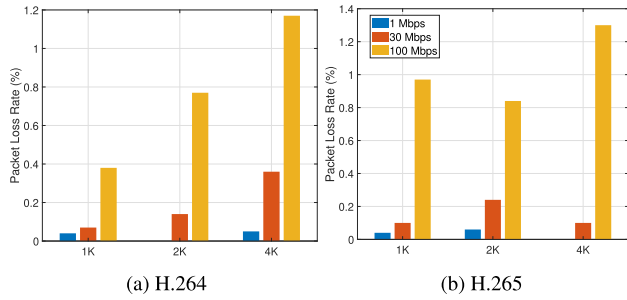


FIGURE 14. Average packet loss summary for VR streams encoded using H.264 (left) and H.265 (right) at different resolutions and bit rates.

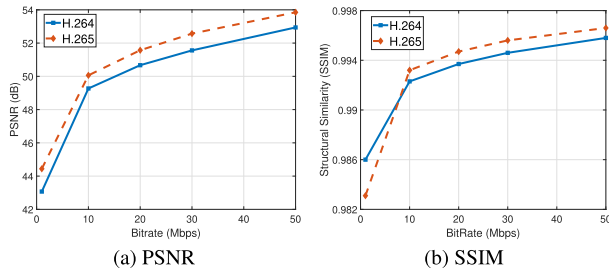


FIGURE 15. Comparison of VR image quality with varying bit rate in the CBR mode.

packet loss rate of H.265 reaches approximately 1%. However, such packet losses lead to a frame rate decrease of only 1 FPS relative to the generated frame rate, as seen in the previous subsection. The packet loss rate for H.265-encoded frames is not much different from that for frames encoded with H.264, especially for a bit rate of 100 Mbps.

C. QUALITY ASSESSMENT METRICS

The PSNR and SSIM evaluation results are presented in Fig. 15. These results show that the graphics quality improves in terms of both metrics when more bits are used to encode the VR video frames. When the bit rate is initially increased from 1 to 10 Mbps, both metrics improve considerably, while the slopes of increase in both metrics become less steep for further increases in the bit rate. It can also be observed that H.265 achieves higher PSNR and SSIM values overall than H.264 for the same bit rate. This can be attributed to the superior compression performance of H.265 given the same number of bits. We can therefore conclude that better VR image quality is attained by encoding streams using higher bit rates. However, encoding streams at higher bit rates also produces larger payloads. Thus, the optimal value of the bit rate should be chosen to strike a balance between acceptable perceived quality and an excessively large frame size.

VII. LATENCY MEASUREMENTS

In our experiments, we measure the total end-to-end latency and the latencies of the new components introduced by VR offloading, as defined below:

- **Total latency (Δt_{total}):** the delay between the time when the pose data are generated and sent by the client device to the server ($t_{\text{pose-data}}$) and the time when the corresponding VR frame is sent to the display of the device for presentation ($t_{\text{display-buffer-in}}$). The total latency of frame i is given as follows:

$$\Delta t_{\text{total}}[i] = t_{\text{display-buffer-in}}[i] - t_{\text{pose-data}}[i]. \quad (2)$$

- **Encoding latency ($\Delta t_{\text{encoding}}$):** the time interval between the time when a frame is read out from the renderer and presented to the encoder ($t_{\text{encoder-in}}$) and the time when the video frame finishes being compressed and is ready for packetization before transmission ($t_{\text{encoder-out}}$). The encoding latency of frame i is given as follows:

$$\Delta t_{\text{encoding}}[i] = t_{\text{encoder-out}}[i] - t_{\text{encoder-in}}[i]. \quad (3)$$

- **Transport latency ($\Delta t_{\text{transport}}$):** the delay between the time when an encoded frame is sent from the application layer of the computing host ($t_{\text{server-app-out}}$) and the time when the frame is successfully received by the application layer of the client device ($t_{\text{client-app-in}}$). The transport latency of frame i is given as follows:

$$\Delta t_{\text{transport}}[i] = t_{\text{client-app-in}}[i] - t_{\text{server-app-out}}[i]. \quad (4)$$

- **Decoding latency ($\Delta t_{\text{decoding}}$):** the delay between the time when a frame is input to the decoder ($t_{\text{decoder-in}}$) and the time when the corresponding decoded frame is output by the decoder to be scanned out ($t_{\text{decoder-out}}$). The decoding latency of frame i is given as follows:

$$\Delta t_{\text{decoding}}[i] = t_{\text{decoder-out}}[i] - t_{\text{decoder-in}}[i]. \quad (5)$$

We first present the time series evolution and CDF curves showing the trends in the individual latency components and the overall total latency in Fig. 16. The curves indicate that the total latency is affected by the encoding bit rate and the resolution. For the 4K resolution, the total latency is approximately 50-60 ms for a bit rate of 1 Mbps but reaches approximately 90 ms for a bit rate of 100 Mbps, peaking at over 100 ms. It is also shown that H.264 has a lower total latency than H.265. Moreover, when similar encoding bit rates are used, the 2K resolution results in smaller delays than the 4K. The causes of the above trends are explained in terms of the encoding, decoding, and transport delays in the following.

The CDF curves for the encoding and decoding latencies under various configurations are presented in Fig. 17. We observe from these curves that, at a similar encoding bit rate, lower-resolution content is encoded and decoded with lower delays than its higher-resolution counterparts. When we keep the resolution fixed, we encounter lower delays for compression and decompression when fewer bits are used for encoding. H.265 consumes longer encoding and decoding times than H.264 due to its heavier computation for obtaining frames of reduced size with superior quality.

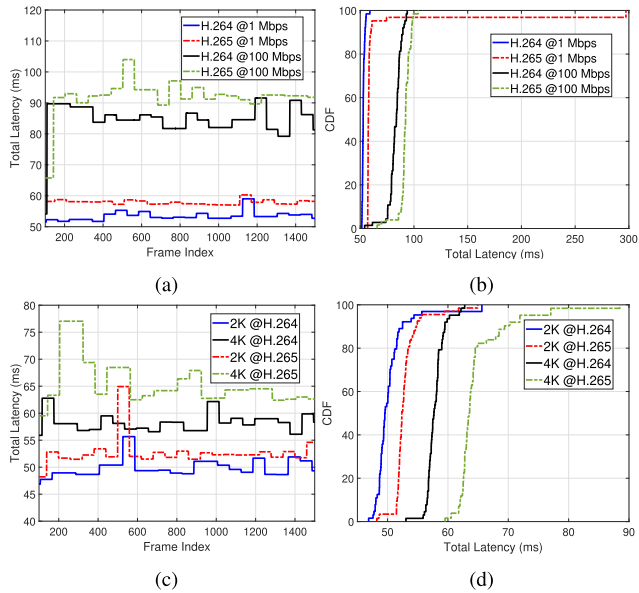


FIGURE 16. System latency patterns under various encoding settings. Trends for VR videos at 4K resolution (a, b) and when the same bit rate (30 Mbps) is used for encoding (c, d)

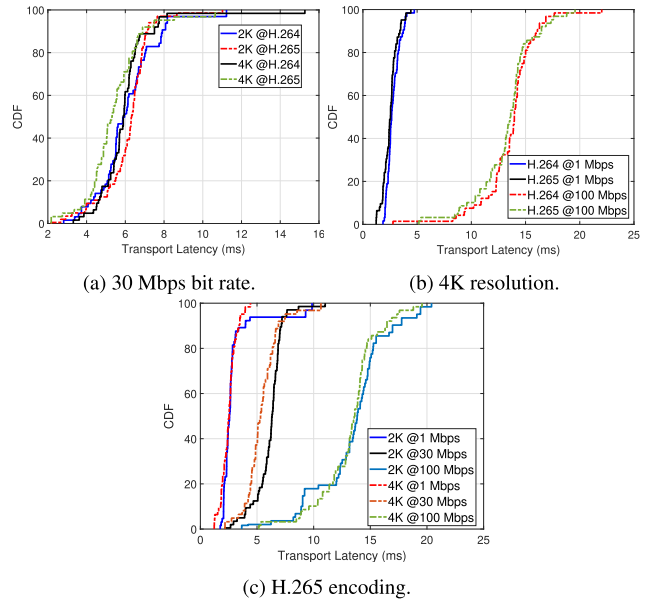


FIGURE 18. CDF curves of transport latency under different encoding parameters.

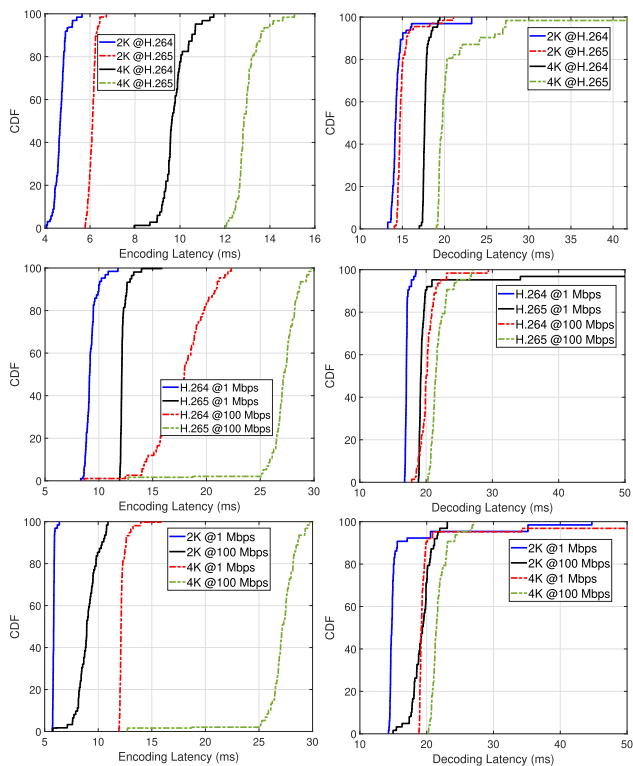


FIGURE 17. CDF comparison of the encoding and decoding latencies when a bit rate of 30 Mbps is used (top), 4K content is streamed (middle), and the H.265 codec is used (bottom), along with various indicated encoding configurations.

The transport latency tends to only be affected by the bit rate settings, as depicted in Fig. 18. As seen in (a), at a bit rate of 30 Mbps, the transport latency is mostly lower than

10 ms for both 2K and 4K video frames encoded with both the H.264 and H.265 encoders. However, as seen in (b) and (c), for a bit rate of 100 Mbps, most latency samples range between 10 and 20 ms regardless of the resolution and the encoder. This implies that when we use a higher bit rate for better image quality, the transport latency becomes a factor influencing the total latency.

The contribution of each latency component to the total latency is indicated in Fig. 19. From these bar charts, we can see that the largest delay originates from the decoding component for both low- and high-resolution contents and at all encoding bit rates, except in the case of H.265 encoding at 100 Mbps. For the 1K resolution, the encoding and transport latencies are comparable to each other, but the transport latency increases slightly at a rate of 100 Mbps. For the 4K resolution, however, the transport latency remains the same as in the 1K case, but the encoding latency tends to increase considerably for both encoders and all data rates.

We observe varying patterns in the encoding latency between the low- and high-resolution cases, whereas the difference in the transport latency between the two resolutions is insignificant. At a high resolution, the encoding latency makes the second highest contribution, while the transport latency makes the lowest. In the case of low-resolution content, the encoding latency makes the lowest contribution of all. We further note from Fig. 19 that the average system latency generally increases with both the resolution and encoding bit rate. Additionally, H.264 achieves lower latency than H.265.

We also provide the latency results of two different host and VR headset hardware platforms in Fig. 20. We denote the host platform used throughout the paper as PC1 (detailed in Section III) and another platform (Intel i7-9750H, 32 GB

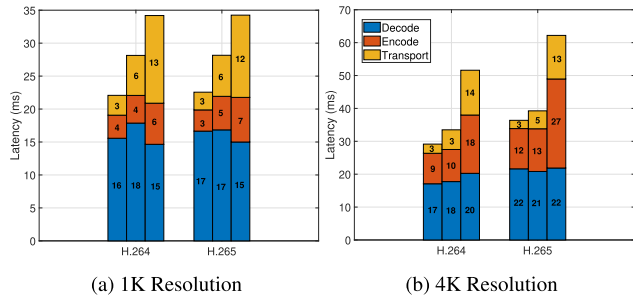


FIGURE 19. Contributions of individual latency components to the total latency when 1K and 4K contents are encoded using different bit rates (left bar: 1 Mbps, middle bar: 30 Mbps, right bar: 100 Mbps).

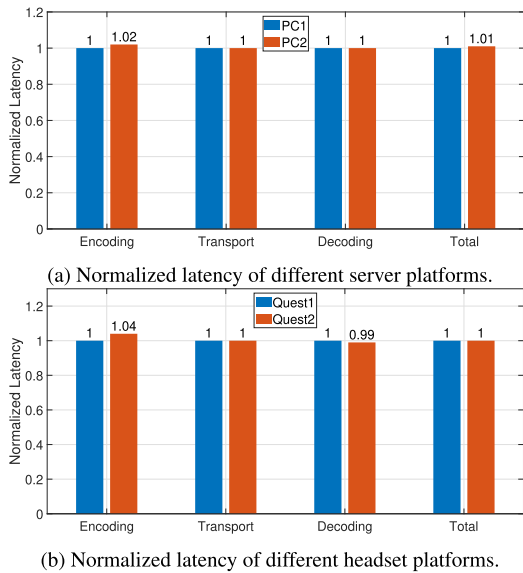


FIGURE 20. Latency comparison between different host and headset hardware platforms.

RAM, RTX 2060 graphics card) as PC2. The results are normalized by those of PC1. The results show that PC2 has a slightly higher encoding latency, though only by 2%. The latency comparison (normalized by the results of Quest 1) between two different headset models, Quests 1 and 2, also shows comparable values between them. These results imply that the VR processing of our testbed is not highly affected by the performance differences of the considered host and VR headset hardware platforms in terms of latency. For a better understanding of the impact of hardware performance on VR processing, a wider range of hardware platforms need to be considered, which remains for future work.

VIII. ANGULAR CHANGES, BLACK BORDERS AND OVERFILLING

In this section, we investigate the impacts of the total latency on angular changes, the formation of black borders after reprojection, and overfilling to solve these problems. In experiments, to emulate the overall latency while reproducing the same motion patterns, we record user motion data

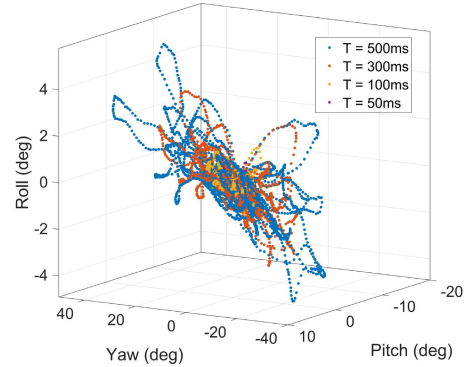


FIGURE 21. Angular changes in 3D space at various latency levels.

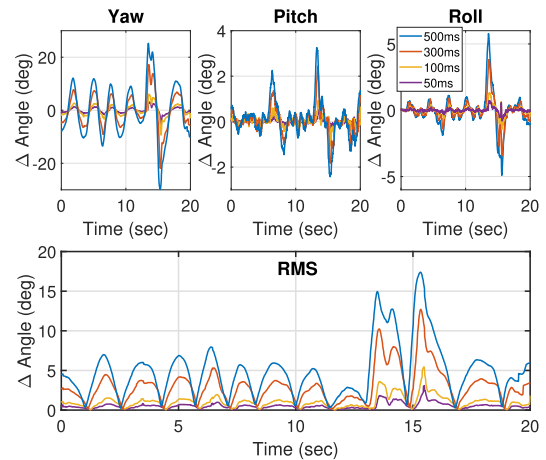


FIGURE 22. Time evolution patterns of angular changes at various latency levels.

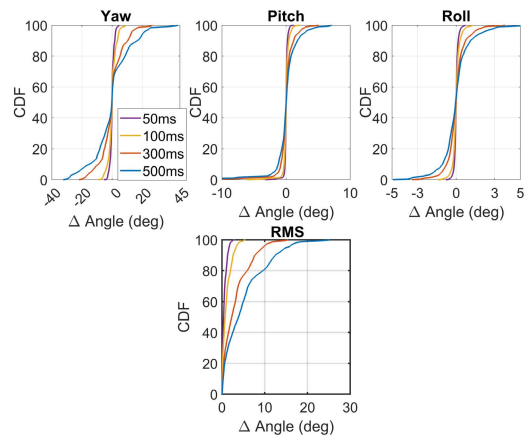


FIGURE 23. CDFs of angular changes at various latency levels.

in a trace file and apply a specified latency to the file. Then, the motion data in the file are streamed to our VR system.

A. LATENCY VS. ANGULAR CHANGES

The angular change between the input motion applied for rendering and the input motion at the time of reprojection

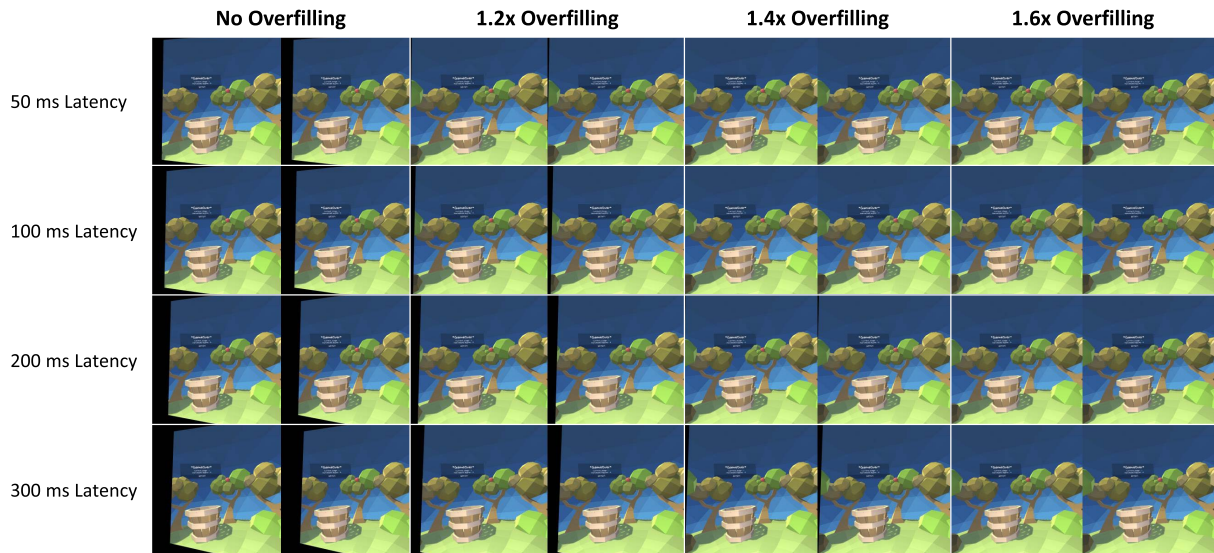


FIGURE 24. Left- and right-eye images of the same VR frame without and with overfilling for various latency values, exhibiting different levels of black borders.

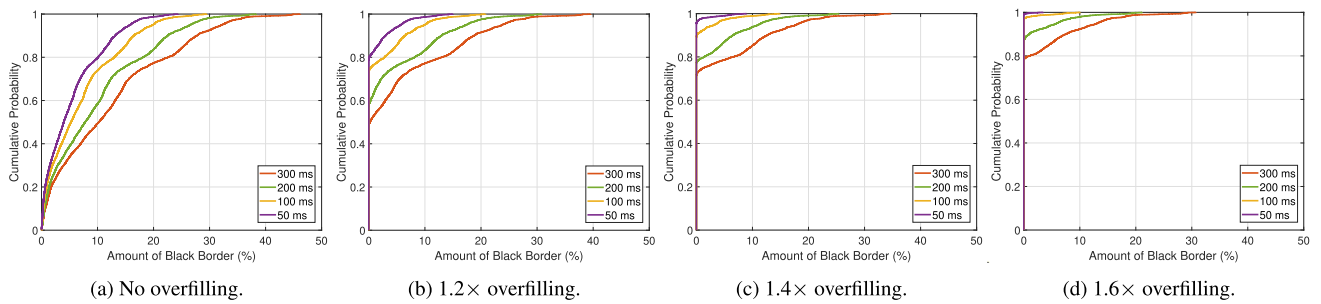


FIGURE 25. CDFs of the black border area for different levels of overfilling applied in various latency cases.

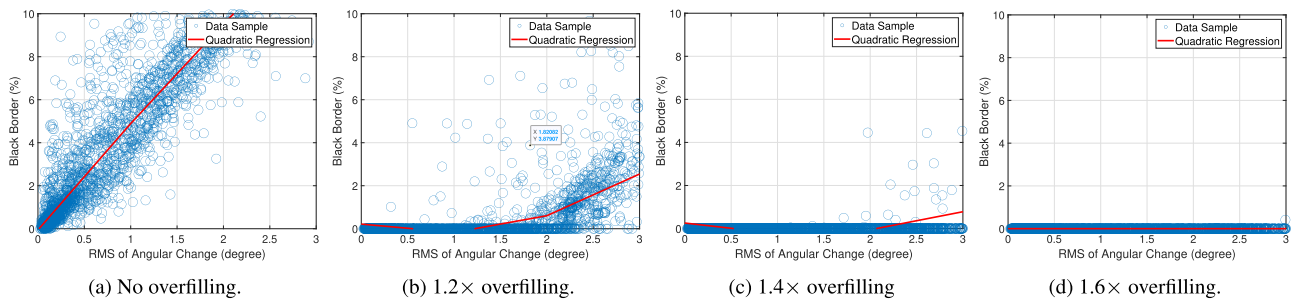


FIGURE 26. Scatter plots of the RMS angular changes vs. black border areas for different levels of overfilling.

strongly affects the quality of a reprojected VR frame. The reprojected VR frame is perceived to be more accurate when the difference is smaller, better approximating the actual situation for certain acceptable error ranges [78]. The difference is what ultimately determines the amount of black border area produced, as is detailed in the next subsection. As observed in the results presented below, the angular change increases as the latency increases. Fig. 21 tracks angular changes for different amounts of latency. As depicted in this figure, the

angular changes extend farther as the latency increases. The time evolution curves of the yaw, pitch, and roll angles and their root mean square (RMS) values in Fig. 22 show that the angular changes are minimal at a latency of 50 ms but gradually increase with an increasing latency, reaching their highest values at 500 ms. The angular changes in the yaw coordinate are the greatest, reaching over 20 degrees, while those in the pitch and roll directions are mostly within 5 degrees. This is further demonstrated by the CDF curves of

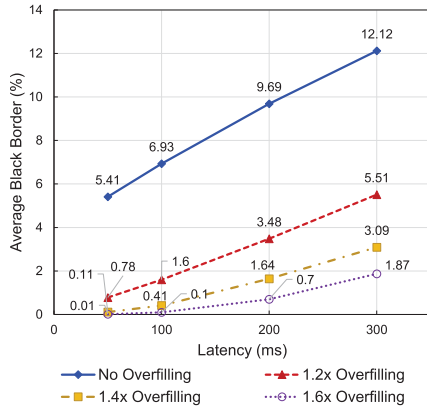


FIGURE 27. Mean black border areas at different latency levels.

the angular changes shown in Fig. 23. If the angular change becomes too large due to high latency and fast user head motion, the difference between the rendered view and the actual view is considerable, resulting in the formation of black borders and consequent deterioration in the viewer’s quality of experience. To correct such errors and repair the resulting black borders, a larger image can be rendered using a technique called overfilling, as detailed in a later subsection.

B. LATENCY VS. BLACK BORDERS

We perform experiments to investigate the effect of the overall latency on the amount of black border area produced. To calculate the black border area ($R_{\text{black-border}}$), the VR system captures a sequence of frame images for a given motion trace. From the obtained images, the number of black pixels ($N_{\text{black-border-pixels}}$) is counted and divided by the number of field-of-view (FOV) pixels (the number of pixels that would fill the screen at zero latency, ($N_{\text{FOV-pixels}}$)) to produce the black border area as a percentage value as given below for VR frame i :

$$R_{\text{black-border}}[i] = \frac{N_{\text{black-border-pixels}}[i]}{N_{\text{FOV-pixels}}} \tag{6}$$

The obtained black border area can range from 0% (no black borders) to 100% (all pixels are black).

Black borders in the same VR reference frame (left- and right-eye images for each frame) under different levels of latency are displayed in Fig. 24. In this figure, we can see that noticeable black borders already appear at the left and bottom sides of the frame at 50 ms of latency, and the black border area gradually increases to reach a substantial amount when the latency is 300 ms.

Such trends are quantitatively captured in the CDFs of the black border area shown in Fig. 25(a) for different latency cases. In this figure, there is a nonzero black border area in the no overfilling case for all VR frames even at 50 ms latency. Moreover, the black border area tends to increase with an increasing latency. Even for a latency of 50 ms, it can be above 20%. For a latency of 300 ms, the average black border

area is approximately 12%, but it reaches up to 45%, meaning that almost half of the screen may appear black. The scatter plot of the RMS angular change vs. the black border area in Fig. 26(a) also shows that the black border area is highly correlated with the angular change.

C. OVERFILLING

Fig. 24 also presents VR image snapshots obtained when more pixels than only those in the FOV are rendered. For example, 1.4x overfilling denotes the case where 40% more pixels than those in the FOV are rendered around the image. These snapshots show that 1.2x overfilling eliminates black borders for a latency of up to 50 ms. However, even with this level of overfilling, black borders remain at higher latency due to the excessive size of the black borders. For 200 and 300 ms of latency, 1.4x overfilling still leaves small black borders. Finally, 1.6x overfilling completely eliminates black borders in all latency cases.

The CDF curves of the black border area for different latency cases are illustrated in Fig. 25. While black borders are seen for all frames in the case of no overfilling, 1.2x overfilling considerably reduces the number of frames with black borders; only 20% of frames have black borders with 1.2x overfilling at 50 ms of latency. However, even with 1.2x overfilling, 50% of frames still have black borders at 300 ms of latency. This is reduced to 30% with 1.4x overfilling and to 20% with 1.6x overfilling, implying that although overfilling is effective in mitigating the black border problem, a higher level of overfilling is needed at higher latency. The scatter plots of the RMS angular changes vs. the black border areas in Fig. 26 show that the slope of the black border area with increasing angular change becomes less steep at a higher level of overfilling. That is, for an equal amount of angular change, a higher level of overfilling results in smaller black borders. This trend is also confirmed by the mean black border areas displayed in Fig. 27. This figure also shows that although the gain from overfilling is higher at a low latency, overfilling is still effective at a high latency. For black border repair, it is better to use the highest degree of overfilling that is feasible. However, rendering more pixels will incur higher computational loads. A tradeoff exists between the severity of the black border phenomenon and the computational load. Therefore, it is desirable to use overfilling to achieve an effective quality increase to an affordable degree within the given budget of computational power. It is also shown that the black border area still increases with an increasing latency for all overfilling cases.

IX. CONCLUSION

In this work, we investigated various characteristics of VR computation offloading through comprehensive experiments based on a prototyped testbed for edge-assisted VR processing and streaming. First, we investigated the benefits of VR offloading in terms of computational load and power consumption reduction for the client device compared to standalone operation. Next, we measured VR traffic patterns

in terms of frame size, data and packet rates under various resolutions and encoding options. We also measure several performance metrics associated with quality of experience, such as frame rate, packet loss rate, and image quality, under various configuration settings. Then, we presented latency measurement studies and investigate the per-component latency under various settings. Furthermore, we studied the impacts of latency and motion patterns on the formation of black borders resulting from image reprojection and the overfilling technique for correcting these black borders. Based on the experimental results, we revealed that VR offloading considerably reduces computational load and power consumption compared to standalone operation, VR traffic patterns, quality and latency are strongly affected by the encoding configuration, the extent of black border formation increases with an increasing latency, and that overfilling successfully reduces black borders but at the cost of increasing computational overhead with an increasing latency.

The experimental study conducted in this paper has several limitations: (1) only a single VR client at a time is considered, (2) a limited set of hardware platforms is considered, (3) wireless connections are not diverse, and (4) positional tracking is not considered. Therefore, reliving the abovementioned limitations, i.e., studying the characteristics of edge VR in more diverse experimental environments, including multiple VR clients at the same time, diverse hosts (e.g., commercial edge and cloud services) and client platforms, wireless connections, such as LTE and 5G NR, and positional tracking remains for future work. The optimization of the VR streaming framework and its communication protocols to minimize the total latency is still an important issue to explore. A holistic/joint design approach of VR content creation, communication protocols and the radio resource management of VR client links will maximize the efficiency of edge VR, and thus, is a subject for future research. Moreover, extending the service platform for augmented and mixed reality will be another important direction for future work.

REFERENCES

- [1] G. Giraldi, R. Silva, and J. Oliveira, "Introduction to virtual reality," LNCC, London, U.K., Res. Rep. 6, 2003.
- [2] D. T. Tan, S. Kim, and J.-H. Yun, "Enhancement of Motion Feedback Latency for Wireless Virtual Reality in IEEE 802.11 WLANs," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.
- [3] A. Mehrabi, M. Siekkinen, T. Kämäräinen, and A. Yi-Jski, "Multi-tier cloudVR: Leveraging edge computing in remote rendered virtual reality," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 17, no. 2, pp. 1–24, 2021.
- [4] C.-Y. Huang, K.-T. Chen, D.-Y. Chen, H.-J. Hsu, and C.-H. Hsu, "GamingAnywhere: The first open source cloud gaming system," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 10, no. 1s, pp. 1–25, 2014.
- [5] HTC. *VIVE wireless Adapter*. Accessed: Sep. 9, 2022. [Online]. Available: <https://www.vive.com/us/accessory/wireless-adapter/>
- [6] Huawei, "Preparing for a cloud AR/VR future (White Paper)," Huawei Technol., Shenzhen, China, Tech. Rep., 2017.
- [7] T. Kämäräinen, M. Siekkinen, J. Eerikäinen, and A. Ylä-Jääski, "CloudVR: Cloud accelerated interactive mobile virtual reality," in *Proc. 26th ACM Int. Conf. Multimedia*, 2018, pp. 1181–1189.
- [8] *NR and NG-RAN Overall Description*, Technical Specification, 3GPP document TR 38.300 V16.8.0, 2021.
- [9] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 8: Enhancements for Extremely High Throughput (EHT)*, IEEE Standard P802.11be Draft 1.0, May 2021.
- [10] M. Abrash, "Latency—the sine qua non of AR and VR," Valve Corp., Bellevue, WA, USA, Tech. Rep., Dec. 2012.
- [11] D. L. David and X. Yang, "EHT Use Case Discussion: VR Requirement Follow up," IEEE Standard 802.11-18/1954r2, Jan. 2019.
- [12] M. Antonov. (2015). *Asynchronous Timewarp Examined*. Developers Blog, Oculus. [Online]. Available: <https://www.oculus.com/blog/asynchronous-timewarp>
- [13] D. Evangelakos and M. Mara, "Extended timewarp latency compensation for virtual reality," in *Proc. ACM SIGGRAPH Symp. Interact. 3D Graph. Games (I3D)*, 2016, pp. 193–194.
- [14] T. C. Nguyen, S. Kim, J.-H. Son, and J.-H. Yun, "Selective timewarp based on embedded motion vectors for interactive cloud virtual reality," *IEEE Access*, vol. 7, pp. 3031–3045, 2018.
- [15] M. Antonov. *Asynchronous Timewarp Examined*. [Online]. Available: <https://developer.oculus.com/blog/asynchronous-timewarp-examined/>
- [16] M. S. Elbambay, C. Perfecto, M. Bennis, and K. Doppler, "Toward low-latency and ultra-reliable virtual reality," *IEEE Netw.*, vol. 32, no. 2, pp. 78–84, Mar. 2018.
- [17] A. Rohloff, Z. Allen, K.-M. Lin, J. Okrend, C. Nie, Y.-C. Liu, and H.-W. Tseng, "OpenUVR: An open-source system framework for untethered virtual reality applications," in *Proc. IEEE 27th Real-Time Embedded Technol. Appl. Symp. (RTAS)*, May 2021, pp. 223–236.
- [18] K.-T. Chen, Y.-C. Chang, H.-J. Hsu, D.-Y. Chen, C.-Y. Huang, and C.-H. Hsu, "On the quality of service of cloud gaming systems," *IEEE Trans. Multimedia*, vol. 16, no. 2, pp. 480–495, Feb. 2014.
- [19] H.-J. Hong, C.-F. Hsu, T.-H. Tsai, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu, "Enabling adaptive cloud gaming in an open-source cloud gaming platform," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 12, pp. 2078–2091, Dec. 2015.
- [20] A. K. Sandhu, "Big data with cloud computing: Discussions and challenges," *Big Data Mining Anal.*, vol. 5, no. 1, pp. 32–40, 2022.
- [21] J. Ren, J. Li, H. Liu, and T. Qin, "Task offloading strategy with emergency handling and blockchain security in SDN-empowered and fog-assisted healthcare IoT," *Tsinghua Sci. Technol.*, vol. 27, no. 4, pp. 760–776, Aug. 2022.
- [22] F. Li, X. Yu, R. Ge, Y. Wang, Y. Cui, and H. Zhou, "BCSE: Blockchain-based trusted service evaluation model over big data," *Big Data Mining Anal.*, vol. 5, no. 1, pp. 1–14, 2022.
- [23] R. Bi, Q. Liu, J. Ren, and G. Tan, "Utility aware offloading for mobile-edge computing," *Tsinghua Sci. Technol.*, vol. 26, no. 2, pp. 239–250, 2021.
- [24] W. Cai, R. Shea, C.-Y. Huang, K.-T. Chen, J. Liu, V. C. Leung, and C.-H. Hsu, "A survey on cloud gaming: Future of computer games," *IEEE Access*, vol. 4, pp. 7605–7620, 2016.
- [25] R. Shea, D. Fu, and J. Liu, "Cloud gaming: Understanding the support from advanced virtualization and hardware," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 12, pp. 2026–2037, Dec. 2015.
- [26] W. Zhang, X. Liao, P. Li, H. Jin, and L. Lin, "ShareRender: Bypassing GPU virtualization to enable fine-grained resource sharing for cloud gaming," in *Proc. ACM Multimedia*, 2017, pp. 324–332.
- [27] R. Shea, D. Fu, and J. Liu, "Rhizome: Utilizing the public cloud to provide 3D gaming infrastructure," in *Proc. ACM MMSys*, 2015, pp. 97–100.
- [28] S. Shi and C.-H. Hsu, "A survey of interactive remote rendering systems," *ACM Comput. Surv.*, vol. 47, no. 4, p. 57, 2015.
- [29] K. Lee, D. Chu, E. Cuervo, J. Kopf, Y. Degtyarev, S. Grizan, A. Wolman, and J. Flinn, "Outatime: Using speculation to enable low-latency continuous interaction for mobile cloud gaming," in *Proc. ACM MobiSys*, 2015, pp. 151–165.
- [30] Z. Lai, Y. C. Hu, Y. Cui, L. Sun, and N. Dai, "Furion: Engineering high-quality immersive virtual reality on today's mobile devices," in *Proc. ACM MobiCom*, 2017, pp. 409–421.
- [31] E. Cuervo, A. Wolman, L. P. Cox, K. Lebeck, A. Razeen, S. Saroiu, and M. Musuvathi, "Kahawai: High-quality mobile gaming using GPU offload," in *Proc. MobiSys*, 2015, pp. 121–135.
- [32] C. Liu, W. T. Ooi, J. Jia, and L. Zhao, "Cloud baking: Collaborative scene illumination for dynamic Web3D scenes," *ACM Trans. Multimedia Comput., Commun., Appl. (TOMM)*, vol. 14, no. 3s, p. 59, 2018.
- [33] K. Boos, D. Chu, and E. Cuervo, "FlashBack: Immersive virtual reality on mobile devices via rendering memoization," in *Proc. ACM MobiSys*, 2016, pp. 291–304.
- [34] *Virtual Reality (VR) Media Services Over 3GPP*, 3GPP document 26.918, Rev. 16.0.0, Dec. 2018.

- [35] *H.265: High Efficiency Video Coding*, ITU-T, Feb. 2018.
- [36] F. Qian, B. Han, L. Ji, and V. Gopalakrishnan, "Optimizing 360 video delivery over cellular networks," in *Proc. ACM ATC*, Oct. 2016, pp. 1–6.
- [37] M. Hosseini and V. Swaminathan, "Adaptive 360 VR video streaming based on MPEG-DASH SRD," in *Proc. IEEE Int. Symp. Multimedia (ISM)*, Jan. 2016, pp. 407–408.
- [38] M. Hosseini, "View-aware tile-based adaptations in 360 virtual reality video streaming," in *Proc. IEEE Virtual Reality*, Mar. 2017, pp. 423–424.
- [39] X. Corbillon, G. Simon, A. Devlic, and J. Chakareski, "Viewport-adaptive navigable 360-degree video delivery," in *Proc. IEEE ICC*, May 2017, pp. 1–7.
- [40] TNO, *Providing VR services through 3GPP Netw.*, 3GPP document Contribution S4-160680, Jul. 2016.
- [41] A. TaghaviNasrabadi, A. Mahzari, J. D. Beshay, and R. Prakash, "Adaptive 360-degree video streaming using layered video coding," in *Proc. IEEE Virtual Reality (VR)*, Mar. 2017, pp. 347–348.
- [42] F. Duanmu, E. Kurdoglu, Y. Liu, and Y. Wang, "View direction and bandwidth adaptive 360 degree video streaming using a two-tier system," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2017, pp. 1–4.
- [43] Z. Xu, X. Zhang, K. Zhang, and Z. Guo, "Probabilistic viewport adaptive streaming for 360-degree videos," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2018, pp. 1–5.
- [44] T. C. Nguyen and J.-H. Yun, "Predictive tile selection for 360-degree VR video streaming in bandwidth-limited networks," *IEEE Commun. Lett.*, vol. 22, no. 9, pp. 1858–1861, Sep. 2018.
- [45] X. Feng, W. Li, and S. Wei, "LiveROI: Region of interest analysis for viewport prediction in live mobile virtual reality streaming," in *Proc. ACM Multimedia Syst. Conf.*, 2021, p. 132–145.
- [46] A. Mehrabi, M. Siekkinen, T. Kämäräinen, and A. Yi-Jski, "Multi-tier CloudVR: Leveraging edge computing in remote rendered virtual reality," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 17, no. 2, pp. 1–24, May 2021.
- [47] J. Wu, C. Yuen, N.-M. Cheung, J. Chen, and C. W. Chen, "Enabling adaptive high-frame-rate video streaming in mobile cloud gaming applications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 12, pp. 1988–2001, Dec. 2015.
- [48] S. Shi, C.-H. Hsu, K. Nahrstedt, and R. Campbell, "Using graphics rendering contexts to enhance the real-time video coding for mobile cloud gaming," in *Proc. 19th ACM Int. Conf. Multimedia (MM)*, 2011, pp. 103–112.
- [49] W. Wei, J. Han, Y. Xing, K. Xue, J. Liu, and R. Zhuang, "MP-VR: An MPTCP-based adaptive streaming framework for 360-degree virtual reality videos," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2021, pp. 1–6.
- [50] S. Chen, B. Duinkharjav, X. Sun, L.-Y. Wei, S. Petrangeli, J. Echevarria, C. Silva, and Q. Sun, "Instant reality: Gaze-contingent perceptual optimization for 3D virtual reality streaming," *IEEE Trans. Vis. Comput. Graphics*, vol. 28, no. 5, pp. 2157–2167, May 2022.
- [51] X. Yang, "Communication-constrained mobile edge computing systems for wireless virtual reality: Scheduling and tradeoff," *IEEE Access*, vol. 6, pp. 16665–16677, 2018.
- [52] T. Dang and M. Peng, "Joint radio communication, caching, and computing design for mobile virtual reality delivery in fog radio access networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 7, pp. 1594–1607, Jul. 2019.
- [53] P. Lin, Q. Song, D. Wang, F. R. Yu, L. Guo, and V. C. M. Leung, "Resource management for pervasive-edge-computing-assisted wireless VR streaming in industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 17, no. 11, pp. 7607–7617, Nov. 2021.
- [54] F. Guo, F. R. Yu, H. Zhang, H. Ji, V. C. M. Leung, and X. Li, "An adaptive wireless virtual reality framework in future wireless networks: A distributed learning approach," *IEEE Trans. Veh. Technol.*, vol. 69, no. 8, pp. 8514–8528, Aug. 2020.
- [55] M. Chen, W. Saad, C. Yin, and M. Debbah, "Data correlation-aware resource management in wireless virtual reality (VR): An echo state transfer learning approach," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4267–4280, Jun. 2019.
- [56] X. Z. Min Huang, "MAC scheduling for multiuser wireless virtual reality in 5G MIMO-OFDM systems," in *Proc. IEEE ICC Workshops*, May 2018, pp. 1–6.
- [57] O. Abari, D. Bharadia, A. Duffield, and D. Katabi, "Enabling high-quality untethered virtual reality," in *Proc. USENIX NSDI*, 2017, pp. 531–544.
- [58] J. Kim, J.-J. Lee, and W. Lee, "Strategic control of 60 GHz millimeter-wave high-speed wireless links for distributed virtual reality platforms," *Mobile Inf. Syst.*, vol. 2017, Mar. 2017, Art. no. 5040347.
- [59] W. Na, N. Dao, J. Kim, E.-S. Ryu, and S. Cho, "Simulation and measurement: Feasibility study of tactile internet applications for mmWave virtual reality," *ETRI J.*, vol. 42, no. 2, pp. 163–174, 2020.
- [60] WILUS and SeoulTech, *Experiments on Wireless VR for EHT*, IEEE Standard 802.11 EHT TIG/SG Contribution 802.11-18/1606r1, 2018.
- [61] J. Ahn and R. Y. K. Young Yong Kim, "Delay oriented VR mode WLAN for efficient wireless multi-user virtual reality device," in *Proc. IEEE ICCE*, Jan. 2017, pp. 122–123.
- [62] D. T. Tan, S. Kim, and J.-H. Yun, "Enhancement of motion feedback latency for wireless virtual reality in IEEE 802.11 WLANs," in *Proc. IEEE GLOBECOM*, Feb. 2019, pp. 1–6.
- [63] Y. M. Saputra and J.-H. Yun, "Motion-aware interplay between wigig and wifi for wireless virtual reality," *Sensors*, vol. 20, no. 23, p. 6782, 2020.
- [64] G. Xiao, H. Li, C. Han, Y. Liu, Y. Li, and J. Liu, "Cloud rendering scheme for standalone virtual reality headset," in *Proc. Int. Conf. Virtual Reality Visualizat. (ICVRV)*, 2020, pp. 316–319.
- [65] *ZeroMQ: An Open-Source Universal Messaging Library*. Accessed: Sep. 9, 2022. [Online]. Available: <https://zeromq.org/>
- [66] D. Wagner, *Motion to Photon Latency in Mobile AR and VR*. Accessed: Sep. 9, 2022. [Online]. Available: <https://medium.com/@DAQRI/motion-to-photon-latency-in-mobile-ar-and-vr-99f82c480926>
- [67] Zman, Guru3, and Lucifer, *Armagetron Advanced*. Accessed: Sep. 9, 2022. [Online]. Available: <http://www.armagetronad.org/>
- [68] *Great Power Demo*. Accessed: Sep. 9, 2022. [Online]. Available: <https://forums.oculusvr.com/developer/discussion/3507/release-great-power/>
- [69] B4TGames, *Epic Roller Coasters*. Accessed: Sep. 9, 2022. [Online]. Available: https://store.steampowered.com/app/787790/Epic_Roller_Coasters/
- [70] HwiNFO64, *HwiNFO—Free System Information, Monitoring and Diagnostics*. Accessed: Sep. 9, 2022. [Online]. Available: <https://www.hwinfo.com/>
- [71] ADPower, *ADPower—HPM-100A Wattman*. Accessed: Sep. 9, 2022. [Online]. Available: <http://shop2.adpower21.com.cafe24.com/>
- [72] E. G. *Showdown VR Demo*. Accessed: Sep. 9, 2022. [Online]. Available: <https://www.unrealengine.com/marketplace/en-US/product/showdown-demo>
- [73] C. GmbH, *Acan's Call: Act 1*. Accessed: Sep. 9, 2022. [Online]. Available: https://store.steampowered.com/app/501180/Acans_Call_Act_1/
- [74] C.-Y. Huang, C.-H. Hsu, Y.-C. Chang, and K.-T. Chen, "Gaminganywhere: An open cloud gaming system," in *Proc. 4th ACM Multimedia Syst. Conf.*, 2013, pp. 36–47.
- [75] ANTYCIP, *The Importance of Frame Rates in VR*. Accessed: Sep. 9, 2022. [Online]. Available: <https://steantycip.com/blogs/the-importance-of-framerates-in-vr/>
- [76] N. Merchant, *The Importance of Frame Rates*. Accessed: Sep. 9, 2022. [Online]. Available: <https://help.irisvr.com/hc/en-us/articles/215884547-The-Importance-of-Frame-Rates>
- [77] *FFmpeg*. Accessed: Sep. 9, 2022. [Online]. Available: <https://www.ffmpeg.org/about.html>
- [78] X. Hou, J. Zhang, M. Budagavi, and S. Dey, "Head and body motion prediction to enable mobile VR experiences with low latency," in *Proc. IEEE GLOBECOM*, Dec. 2019, pp. 1–7.



BARAKA WILLIAM NYAMTIGA received the B.S. degree in computer science from the University of Dar es Salaam, Tanzania, and the M.S. degree in information and communication science and engineering from The Nelson Mandela African Institution of Science and Technology, Arusha, Tanzania. He is currently pursuing the Ph.D. degree in electrical and information engineering with the Seoul National University of Science and Technology, Seoul, South Korea. His current research interest includes virtual reality offloading.



AIRLANGGA ADI HERMAWAN received the B.S. degree in electrical engineering from Gadjah Mada University, Yogyakarta, Indonesia, and the M.S. degree in computer science and engineering from the Technical University of Eindhoven, Eindhoven, The Netherlands. He is currently pursuing the Ph.D. degree in electrical and information engineering with the Seoul National University of Science and Technology, Seoul, South Korea. His current research interest includes virtual reality offloading.



DEOK-YOUNG JUNG received the bachelor's and master's degrees as a major in communication design from the College of Fine Arts, Hongik University, Seoul, South Korea, in 2000 and 2003, respectively. In 2002, he worked with Bbox studio, a computer graphics studio for movies and commercial films. He won the Special Effects Award at the Daejeong Film Festival in South Korea for the film "Faceless Beauty." In 2004, he founded Computer Graphics Studio B1 for movies and games. In 2007, he established Studio Varsia, a mobile game developer. In addition, he established Clicked Inc., in 2013, where he is currently the CEO. Clicked is currently focusing on developing a number of XR contents and WiFi-based XR cloud streaming solutions.



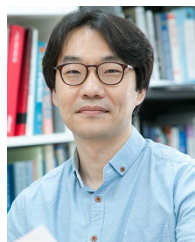
YAKUB FAHIM LUCKYARNO received the B.S. degree in physics from Gadjah Mada University, Yogyakarta, Indonesia, and the M.S. degree in computer engineering from the King Mongkut's Institute of Technology Ladkrabang, Bangkok. He is currently pursuing the Ph.D. degree in electrical and information engineering with the Seoul National University of Science and Technology, Seoul, South Korea. His current research interest includes virtual reality offloading.



JIN SAM KWAK received the B.S., M.S., and Ph.D. degrees in electrical engineering and computer science from Seoul National University, Seoul, South Korea, in 1998, 2000, and 2004, respectively. From 2004 to 2005, he was a Postdoctoral Research Associate with the School of Electrical and Computer Engineering, Georgia Institute of Technology. In 2006, he was with the Wireless Networks and Communications Group (WNCG), The University of Texas at Austin, as a Postdoctoral Research Fellow. From 2007 to 2012, he was working as a Chief Research Engineer with LG Electronics. During this time, he carried out research tasks on the IMT-Advanced & its beyond and led the standards activities for wireless communications in IEEE 802 (especially IEEE 802.11/15/16/19), Wi-Fi Alliance (served as an alternative board member), and WiMAX Forum. Since 2012, he has been working with the WILUS Institute of Standards and Technology, where he is currently the CEO and the Co-Founder. His main research interests include next generation standards-driven technologies for LTE, (B)5G, Wi-Fi, and multimedia codec.



TAE-WOOK KIM received the B.S. degree in computer science from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2007. He is currently a Software Engineer with Clicked Inc., Seoul, South Korea, and his current work focuses on real-time graphic rendering and streaming for virtual reality.



JI-HOON YUN received the B.S. degree in electrical engineering and the M.S. and Ph.D. degrees in electrical engineering and computer science from Seoul National University (SNU), Seoul, South Korea, in 2000, 2002, and 2007, respectively. He was a Senior Engineer with the Telecommunication Systems Division, Samsung Electronics, Suwon, South Korea, from 2007 to 2009. He was a Postdoctoral Researcher with the Real-Time Computing Laboratory, The University of Michigan, Ann Arbor, MI, USA, in 2010. He is currently a Professor with the Department of Electrical and Information Engineering, Seoul National University of Science and Technology (SeoulTech), Seoul. Before joining the SeoulTech, in 2012, he was working with the Department of Computer Software Engineering, Kumoh National Institute of Technology, as an Assistant Professor. His current research interests include wireless communications and networking and mobile applications.

...