Experimental Evaluation of Virtual Reality Applications Running on Next-Gen Network Scenarios with Edge Cloud Assistance

Shalini Choudhury
shalini@winlab.rutgers.edu
WINLAB, Rutgers University
North Brunswick, New Jersey, USA

Jakub Kolodziejski
jkol@winlab.rutgers.edu
WINLAB, Rutgers University
North Brunswick, New Jersey, USA

Ivan Seskar
seskar@winlab.rutgers.edu
WINLAB, Rutgers University
North Brunswick, New Jersey, USA

Dipankar Raychaudhuri
ray@winlab.rutgers.edu
WINLAB, Rutgers University
North Brunswick, New Jersey, USA

ABSTRACT

Volumetric video is an emerging key technology for immersive representation of 3D spaces and objects. Rendering volumetric video at client's end requires significant computational power which is challenging especially for mobile devices. One of the ways to mitigate this is to offload the rendering at the edge cloud and stream the video and audio to the thin client. Remote edge-cloud rendering may increase the end-to-end delay of the system due to the added network and processing latency which is greater than local rendering system. We investigate network latency in edge-based remote rendering over NextG networks and identify the bottleneck deteriorating the application performance. Further, we delve into the current state of the art and challenges of performing rendering remotely at the edge cloud and study the associated problems that need to be addressed in order to realize remote augmented reality(AR)/virtual reality (VR). Our prototype implementation shows effectiveness of maintaining the application QoE by prioritizing data at the level of a sub-flow and reducing the motion-to-photon latency.

WiNTECH '22, October 17, 2022, Sydney, NSW, Australia

ACM ISBN 978-1-4503-9527-4/22/10…$15.00
https://doi.org/10.1145/3556564.3558235

CCS CONCEPTS

• **Networks** → **Network performance evaluation**; **Network experimentation**; • **Computer systems organization** → *Real-time systems.*

KEYWORDS

Virtual Reality, edge-computing, remote rendering, latency, experiments, wireless networks and communications

ACM Reference Format:
Shalini Choudhury, Jakub Kolodziejski, Ivan Seskar, and Dipankar Raychaudhuri. 2022. Experimental Evaluation of Virtual Reality Applications Running on Next-Gen Network Scenarios with Edge Cloud Assistance. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (WiNTECH '22).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3556564.3558235

1 INTRODUCTION

AR/VR technologies are gaining momentum in view of the anticipated impact on various industries including education sector [7, 20], medical training [18], military operation [11] and entertainment [15] and smart city [4, 13] to name a few. There have been considerable advances in both the AR/VR technology domains as well as platform network and computing technologies, and these advances have help create new solution verticals bringing additional immersive experiences at the client's end. Virtual Reality (VR) is an artificial rendering of an environment making use of audio and visual fields possibly supplemented with other sensory devices [1]. Traditionally, rendering can be performed at the client's side leveraging a compute platform connected to the VR headset systems such as Oculus Rift or HTC Vive [10]. In this scenario a high-quality graphics is achieved when the head-mounted display (HMD) is tethered to a powerful GPU,

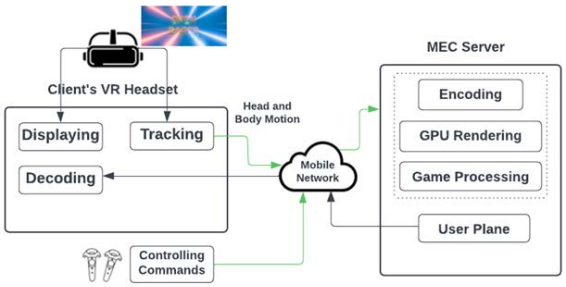

Figure 1: VR remote rendering at the edge cloud via mobile network

thus, limiting the users' mobility to narrow head movement. Untethered HMDs, such as the Samsung Gear VR or Google Cardboard, allow the user to freely move around while wearing the device, but they do not provide translational position tracking and the mobile GPUs force design compromises on rendering quality. However, it is possible to improve the graphics quality on low powered, untethered HMDs by offloading the rendering task to a remote compute platform [21]. Remote rendering serves audio and video content from a VR application to a client, encodes the rendered frames and system audio and sends the frames and audio files to the client for decoding and display. The server also accepts motion coordinates and other control data from the client device as shown in figure 1. There is a stringent low latency requirement for remote rendering in order to provide an acceptable user experience. A popular application of remote rendering is cloud gaming and depending on the type of game its latency requirement typically ranges between 60 and 120ms (only includes game action) [8]. To address the low latency VR gaming requirement, the remote rendering was performed at the Mobile Edge Cloud (MEC).However, even rendering at MEC couldn't comply to the low latency requirement of the VR applications.

2 BACKGROUND AND RELATED WORK

2.1 Capacity Limitation

Current 5G system design efforts aim to support the enormous growth in data rate requirements from resource-hungry applications [19]. This will be realized through increased bandwidth and improved spectral efficiency. Considering VR technology, a study [2] reveals that with each human eye being able to see up to 64 million pixels at a certain moment, and up to 15.5 billion pixels/s are needed to experience real-like view. To be able to facilitate such an experience a required bit rate of up to 1 Gb/s is a necessity. The values above are clearly unrealizable with the current state-of-the-art access network architecture. Additionally, to address the requirement of real-time response for dynamic and interactive collaborative VR applications, there is a significant ongoing research effort aimed at reducing bandwidth needs in mobile VR. In a previous study [16] in the context of 360° immersive VR video streaming, head movement prediction is used to spatially segment raw frames and only deliver their field of view (FOV) portion in HD. Similarly in [9], splitting the video into separate grid streams and serving grid streams corresponding to the FOV is attempted. Such a foveated rendering approach reduces the data rate requirement to about 100 Mb/s for a VR system with less than 10 ms round trip time including rendering in the cloud. However, even if we allow only 5 ms latency for generating a foveated 360° transmission, existing networks cannot serve 100 Mb/s to multiple users VR application with reliable round-trip times of less than 5 ms. Thus, there is a gap between the network requirement of a VR application which necessitates careful design of the NextG (5G and beyond) access.

2.2 Latency Constraint

VR applications demand stringent latency requirements to be able to provide a pleasant immersive experience. The human eye needs to perceive accurate and smooth movements with motion-to-photon (M2P) latency of less than 20ms [6] to avoid motion sickness. However, with the state-of-the-art network the M2P requirement is violated. The challenge for bringing M2P latency down to acceptable levels starts by first understanding the various delay components that contribute to the latency budget as shown in figure 2. Delay contributions to the end-to-end wireless mobile VR latency include sensor detection and action capture; computation, rendering and encoding; framing and streaming; network transport; terminal decoding; and screen refresh. Sensor delay contributes less than 1 ms, and display delay is expected to drop to 5 ms [14], which leaves 14 ms as delay budget for computing and communication. Both computing and communication delay serve as delay bottlenecks in VR systems. Heavy image processing requires high computational power that is often not available in the local HMD GPUs. Offloading computing tasks for remote rendering relieves the computing burden from the users' HMDs at the expense of incurring additional communication delay in both directions. Current communication delay (edge of network to server) can reach 40 ms [5], which clearly exceeds the less than 20ms MTP latency budget.

2.3 Reliability Requirement

VR applications need to consistently meet stringent latency and reliability constraints. Packet drops need to be kept to a

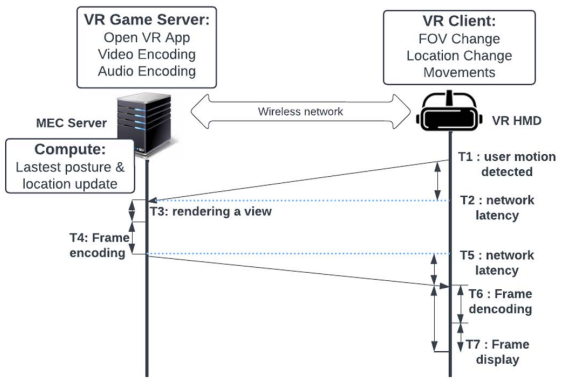

Figure 2: Components of motion-to-photon (M2P) latency: user motion is detected at T1, after T2 network delay, the user motion reaches server, and the rendering occurs for time T3, followed by frame encoding T4. Incurring network delay T5, encoded frames reaches the client and frame decoding happens for time T6 and displayed after T7.

minimum for enhanced user experience. In order to be able to deliver degradation-free Immersive VR experience, error robustness guarantee in different layers, spanning from the video compression techniques in the coding level to the video delivery schemes in the network level is mandatory. This can be guaranteed with ultra-reliable VR service, but the fact is that enhancing reliability always comes at the price of using more resources and may result in additional delays. Another important reliability aspect in current state-of-the-art network is that a maximum packet error rate (PER) of 10−5 is specified in the 3GPP standard. This correlates with the VR/AR tracking messages that have to be delivered with ultra-high reliability to ensure smooth VR service.

Previous studies explore solutions to improve the performance of the current VR system [2, 9, 16]. To provide high-quality VR on a mobile device, [3] presents a pre-rendering and caching design. Parallel rendering and streaming mechanism are adopted to reduce the add-on streaming latency, by pipe lining the rendering, encoding, transmission, and decoding procedures [12]. This work [10] presents a collaborative rendering method to reduce overall rendering latency by offloading to an edge computing node. Clearly, in the current state-of-the-art situation ultra-low latency VR application not only relies on video frame pre-rendering, viewpoint prediction or parallel rendering but also relies on network latency budget in order to address the low end-to-end M2P latency requirement. The latency budget for the round-trip network transport delay in M2P is considered to be around 5 to 7ms [14]. However, the current network architecture is unable to provide the higher bit rates and deliver less than 20ms M2P latency.

As part of our efforts we address some of the challenges that have been identified above in this work, including:

- Building a framework which supports and facilitates experimentation to study the components of network latency and identify the network parameters degrading VR application performance.
- Showcasing the framework's capabilities by designing an experiment where the VR application traffic subflow was prioritized to achieve gain in quality-of-experience (QoE).
- Development of a framework allowing VR tracking messages' reliable delivery over TCP, while streaming the audio and video frames over UDP to reduce resource consumption and adhere to the latency budget for smooth VR service.

To provide a description of the framework and experimentation and analyze the impact of network parameters on the VR application, this paper is organized as follows: Section 3 discusses the system architecture of the VR prototype setup, Section 4 provides a detailed overview of how the measurements were conducted, results and discussion is given in Section 5, future experimentation is discussed in Section 6 and finally Section 7 concludes the paper.

3 SYSTEM ARCHITECTURE

In this section the prototype framework is discussed, to enable several experiments for emulating diverse network conditions and understanding how network conditions affect the performance of remotely rendered VR game application. A simplified version of this architecture is shown in figure 3.

3.1 MEC server architecture

The server is a windows 10 machine backed by NVIDIA Virtual GPU technology. NVIDIA's CloudXR is used for streaming virtual reality from any OpenVR application on the server to the client device such as a VR head-mounted display. The CloudXR presents a virtual HMD driver to SteamVR on the server side. This allows existing OpenVR applications to see the HMD as being locally connected, and thus does not require any changes at the application level. The CloudXR server receives audio and video frames from the OpenVR application and subsequently encodes and transports them to the CloudXR client. The server also accepts motion and control data from the client device. The application that will be streamed on the client's side is launched on the server and the application typically used for this work was a VR game named Open Saber discussed later in this section .

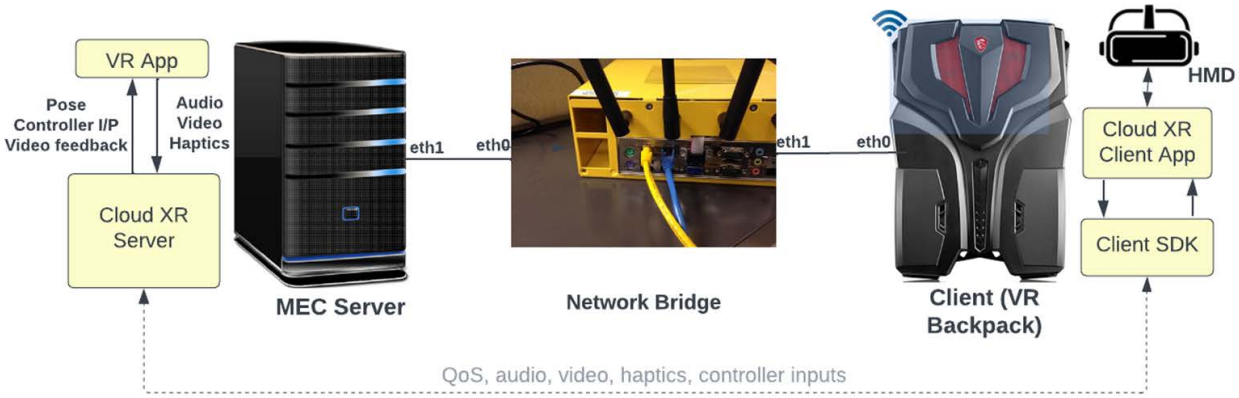

Figure 3: VR Prototype System Setup

3.2 Client architecture

The Windows 10 client decodes and renders content that is sent from the server and collects motion and controller data that is sent to the server. The client is connected to an HTC Vive HMD, a pair of controller to track client motions and has an installation of Steam and SteamVR tools. When the client first connects to the server it reports its specifications such as refresh rate and resolution to the server which is then provided to the open VR application. Software development kit (SDK) at the client's end streams graphics-intensive VR content by accessing the graphics server and streaming over a radio signal to the thin client. The SDK enables mobile access to graphics-intensive applications on relatively low-powered graphics hardware.

3.3 Setting up the network

The Server is on the wired network and the client is untethered and wirelessly connected. To avoid packet drops and ensure low latency the client uses Dynamic Frequency Selection (DFS) channel for the wifi network. DFS is a WiFi function that enables WLANs to use 5 GHz frequencies. One main benefit of using DFS channels is to utilize under-serviced frequencies to avail less interference and better wifi performance. A channel availability check was executed during the boot process of the access point (AP) and there after Channel 100 with associated frequency of 80 MHz was selected. With this approach a link-speed of 800 Mbps was achieved between the VR server and client. Close enough to the 5G network's 1 Gb/s bitrate requirement and can achieve peak frame rate of 90 frames per second (fps) for the Open Saber VR game. We initially setup an efficient network that would support enhanced gaming experience. Following which network parameters are varied parametrically one at a time to identify key influence factors for application QoE

3.4 Portable COSMOS Node: Access Point

As seen from figure 3 the yellow box is a COSMOS testbed [17] node which is positioned single hop away from both the VR Client and MEC server. The COSMOS node acts as a bridging device and AP through which the client traffic is steered towards the game server and includes Ethernet connectivity to introduce network fluctuations beyond what a single hop network would experience. This COSMOS node is imaged with network emulation (NetEm) functionalities to achieve traffic control facilities and add or reduce delay, packet loss, packet reordering and other characteristics to traffic outgoing from the client. As shown in figure 3 the traffic outgoing from eth0 interface of the COSMOS node has injected NetEm functionalities.

3.5 Game Selection

We carefully selected the following two representative VR games for our experiments

- **Open Saber** is a rhythm VR music game. In this game, players slash the boxes, which are flying toward them. We chose Open Saber because it is a latency-sensitive game with faster pace. We conjecture that its gaming experience is affected by the latency
- **Together VR** is a leisure interactive game. In this game, players experience everyday life with an avatar. This game has the slowest pace with simple game scenes

  The experimental measurements presented in this paper is based on Open Saber game since it is complex

in both spatial information and temporal information domain.

4 MEASUREMENT OVERVIEW

The majority of the end-to-end latency comes from the network [22]. Additionally, the network will be strained even more, if it has to support high resolution HMDs in the future. Hence, in this section we conduct several experiments emulating diverse network conditions to understand how network variability affect the objective performance of the VR gaming. The network parameters varied in the gaming prototype are listed in table 1. Table 2 provides baseline measurements of Open Saber VR game quality-of-service (QoS) parameters. The baseline measurements were conducted while playing the game over public wifi and then over DFS wifi. The QoS parameter measurements over public wifi gives a clear idea that the Open Saber VR game was almost unplayable. However, an efficient QoE was achieved when the game was played over the DFS wifi.

In order to evaluate how network parameters influence VR application QoE, we start regulating the parameters at the AP. The Cloudxr sdk application at the clients end record runtime QoS statistics. The API call to log the QoS parameters is executed by providing launch options from the client console and the measurements are accessed from the client's end.

In our prototype implementation given in figure 3, one network parameter is adjusted at the AP one at a time, for instance when varying the bandwidth, we do not add additional delay nor inject packet losses. As seen from table 1 we vary the following parameters throughout our measurements:

- **Bandwidth**: The bandwidth (BW) is varied from 400 - 900 kbps during the game run-time. The effect of rate limiting impact on QoS parameters including frames per second (fps), average video bit rate, frame time CXR and received packets are recorded and the results are discussed in section 5.
- **Packet Loss & Packet Re-ordering Rate**: We vary the packet loss and reordering rate in the rage 2%, 4%, 6%, 8%, 10%. We drop and reorder packets in only one direction, specifically when packets being sent from the client to server.
- **Delay**: In our experiments one-way delay is varied between 0 - 500 ms. Similar to packet loss rate, the delay is also injected at the AP when traffic is sent from the VR client to the server.

5 RESULTS AND DISCUSSION

The results obtained by varying the network parameters in the prototype framework is discussed in this section:

Table 1: Network Parameter Varied in the Experiments

Parameter	Variation Value
Delay	0 - 500ms
Packet drop	1-10%
Packet reordering	1-10%
Bandwidth	400-900kbps

Table 2: VR game baseline performance over different network

QoS Parameter	Public Wifi	DFS Wifi
Average Video Bitrate (kbps)	525	3200
Frames per sec (fps)	29-31	90
Frame Time (ms)	62	34
M2P Latency (ms)	163	70
Packet Re-Tx (%)	8.1-8.7	0

5.1 Frame rate and frame time are sensitive to uplink network bandwidth bottleneck

Varying the network bandwidth in the range 400 to 900 kbps, we evaluate the effect on the following VR game application parameters:

Video bitrate - Videos to stream steadily, the video resolutions have to match with the right video bitrate

Frame rate - Frame rate also expressed in frames per second (FPS) is the frequency at which consecutive frames are captured or displayed.

Frame time - It is the one way latency from when the player movement leaves the client library to when it is received at the server.

In figure 4 it is seen that a bandwidth bottleneck (rate limiting) is injected in the network for uplink control traffic (control requests and client motions). Due to this limitation, the average video bitrate at the client's end initially is 750 kbps, which corresponds to medium quality video with resolution of 640 x 360. However, when the bottleneck is relaxed beyond 600 kbps, 80% gain is observed in average video bitrate and there is no further improvement in both the QoS parameters (received packets and avg video bitrate). From figure 4a it can be concluded that the maximum average video bitrate required for the Open Saber game is 3200 kbps which corresponds to HD video quality with 1280 x 720 resolution. From figure 4b we notice an interesting phenomenon that when bandwidth bottleneck (rate limiting) is injected for uplink control traffic the frame rate is less than 78 fps. A substantial improvement in fps is seen as we keep increasing the bandwidth, however after 600 kbps the fps shows

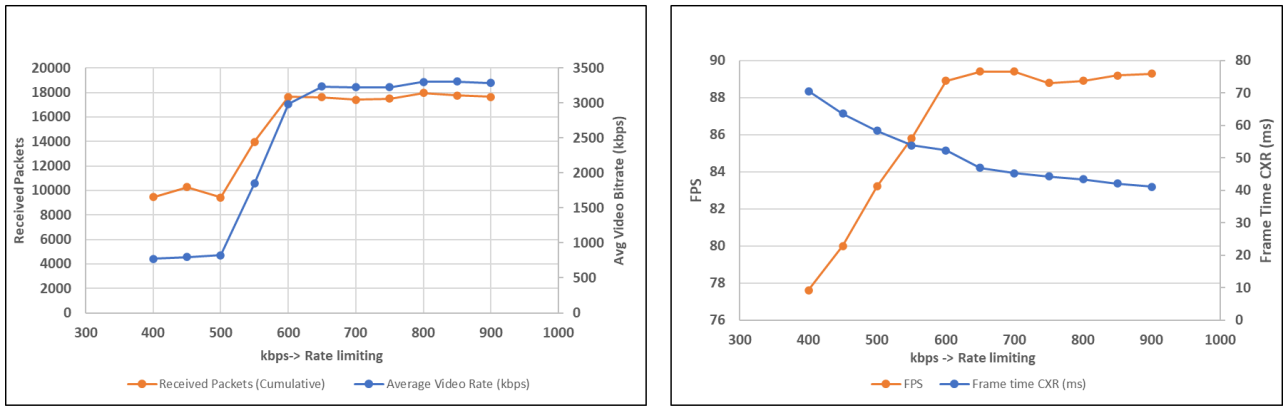

Figure 4: Effect of Bandwidth Bottleneck on a)Left:Avg video bitrate & Received packets - b)Right:Frames per second & Frame Time

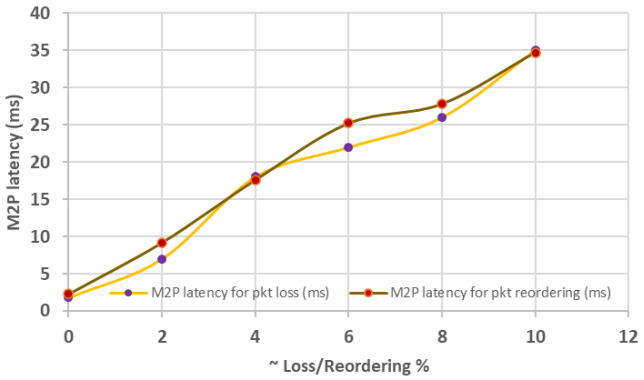

Figure 5: Effect of Packet loss/reordering on Motion-to-Photon Latency

no further improvement. The frame time (client to server delay) gradually decreases with increase in the uplink bandwidth and after 700 kbps the frame time stabilizes at 40ms. Thus, a significant observation made from this experiment is that the uplink bottleneck has an impact on the frame time latency which effects the number of frames (fps) received at the clients due to delayed rendering of audio and video frames at the VR game server. However, the average video bitrate downlink is not significantly impacted by the uplink bottleneck.

5.2 Increase in latency budget due to network packet loss and re-ordering

Figure 5 compares M2P latency for packet loss and packet reordering events. In our prototype packets are dropped during data transmission from the client to the server. Similarly, packet reordering is a common phenomenon in the internet and it is evident from figure 5 that it impacts M2P latency either more or equal to packet loss phenomenon because reordering not only affects the performance of the network but also the packets receiver.

Another observation that can be drawn comparing figure 4b and figure 5 is that the bandwidth bottleneck effects VR gaming latency way more significantly than packet loss/reordering events.

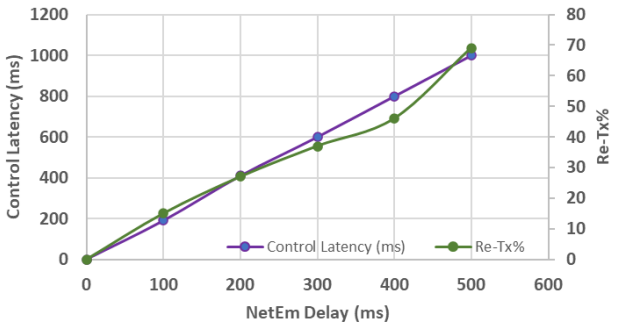

Figure 6: Effect of delay on control latency & retransmission

5.3 Effect of packet retransmission on control latency and streaming latency

The relation between injected NetEm delay and its effect on control and streaming latency is shown in figure 6 and 7. Before discussing the control and streaming latency, it is noted that VR gaming uses real time streaming protocol (RTSP) as the application-level network protocol that

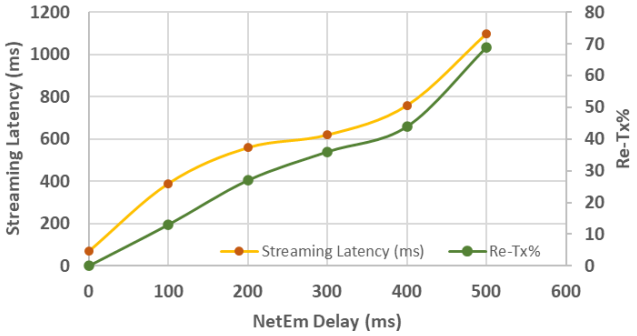

Figure 7: Effect of delay on streaming latency

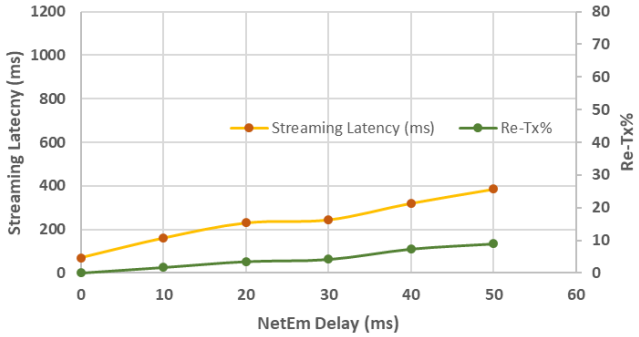

Figure 8: Effect low latency control flow and reduced retransmission rate on streaming latency

transfers real-time data to and from client device by communicating directly with the server streaming the VR game. RTSP operates over user datagram protocol (UDP) when streaming real-time data like game audio and video traffic. The control traffic (includes control request operations and the HMD/controller motions) is sent over transmission control protocol (TCP). In this experiment as we inject delay over the control traffic flow, the current value of TCP retransmission timer is violated and TCP times out and retransmit the outstanding segments to ensure reliability. This event leads to increase in % retransmission depicted by the green curve in figure 6. Hence, as expected both the % retransmission and control latency share a linear relationship with the delay parameter. However, an interesting observation is noted in figure 7 where the injected delay increases % retransmission and a similar trend is observed in the streaming latency curve even though the audio and video streams are sent over connectionless UDP. This phenomenon can be explained with the help of figure 2. The client head and controller movements are sensed and sent to the MEC game server and frames are updated in response to the user interaction. This event spans over time T1 + T2 + T3 + T4 as shown in figure 2. Considering delay is injected in the path of control traffic, the latency component T2 surges, impacting T3 and T4. Thus, resulting in delayed VR audio and video frames update at the VR game server. Specifically, the latency budget for sending the client motion and processing VR content based on client's motion request in the server is exceeding, adding to the streaming latency from server to the client. Thus, similar trend in streaming latency and % retransmission is observed on varying network delay.

5.4 Prioritizing control flow to achieve reduced retransmission and latency

From the previous subsection we took cognizance of the fact that increase in network delay has a positive correlation with control latency and the rate of retransmission. Additionally, the rate of increase in retransmission is same as rate of increase in audio and video streaming latency. Hence, in this experiment an attempt was made to reduce the injected delay on the control traffic and study the behaviour of the streaming latency. Injecting reduced delay in the control traffic flow depicts that the control flow is scheduled on a low latency network slice (figure not included due to space constraint). This approach reduces retransmission rate and as seen from figure 8, this retransmission rate reduction also reduces the streaming latency of the VR game by quick update of client motion on the game server. This allows us to conclude that by prioritizing a traffic sub-flow, we can achieve reduced Motion-to-Photon latency and thus maintain the application QoE.

6 FUTURE EXPERIMENTATION

Our future work will focus on implementing VR applications over a 3GPP compliant standalone 5G network. The experiment will be setup on COSMOS testbed [17] with Open Radio Access Network (O-RAN), CORE, COTs UE enacting as the VR client. COSMOS testbed will support prototyping platform for 5G and comprises of the 5G-NR protocol stack, including standard-compliant implementations of both gNB and UE. Initially, our effort will be to perform an interoperability test of 5G NR with a COTs VR client and then proceed with quantifying and benchmarking performance of VR applications. Key performance parameters of 5G network - resource utilization in time, in frequency, the modulation and coding scheme (MCS), and the transmit and receive gains of the RF front-end, will be regulated to obtain the effect of the system throughput. Throughput and latency profiling for an application involves numerous parameters which we will aim to achieve through a practical system setup in COSMOS 5G testbed for our future work. In the future work we aim to evaluate the system based on multiple clients accessing the network and setting up multiple PDU sessions

simultaneously. To further enable traffic sub-flow priority over a standard complaint 5G network, COSMOS platform supports UE with multiple slices. The network slices will be defined within a public land mobile network (PLMN) and it will incorporate the RAN and Core components. Thus, we will assign different traffic sub-flows of the VR application over multiple slices to maximize the QoS requirement. For example we can enable control traffic flow over ultra-reliable low latency communication (URLLC) and the data traffic flow over enhanced mobile broadband (eMBB) slices.

7 CONCLUSION

This work highlights the challenges degrading VR application performance and analyses the QoS parameters effected by network variability. Even though the rendering is offloaded to the edge of the network, achieving M2P latency of less than 20ms is not practical with current state-of-the-art network. A VR game prototyping framework has been introduced where initially the network supports efficient VR gaming experience but gradually network parameters are adjusted one at a time at the access point to identify the key network parameters that influence the VR application QoE. Further, from the experiments conducted it was concluded that if the control traffic sub-flow was prioritized and scheduled over a low latency network slice the VR game streaming latency can be significantly reduced to approach the desired M2P latency.

ACKNOWLEDGMENTS

This work was supported in part by a research gift from Accenture. We would like to thank Dr. Sanjoy Paul, Accenture Technology Labs, for his comments and thoughtful suggestions in this project

REFERENCES

[1] Christoph Anthes, Rubén Jesús García-Hernández, Markus Wiedemann, and Dieter Kranzlmüller. 2016. State of the art of virtual reality technology. In *2016 IEEE aerospace conference*. IEEE, 1–19.

[2] Ejder Bastug, Mehdi Bennis, Muriel Médard, and Mérouane Debbah. 2017. Toward interconnected virtual reality: Opportunities, challenges, and enablers. *IEEE Communications Magazine* 55, 6 (2017), 110–117.

[3] Kevin Boos, David Chu, and Eduardo Cuervo. 2016. Flashback: Immersive virtual reality on mobile devices via rendering memoization. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*. 291–304.

[4] Joel Carneiro, Rosaldo JF Rossetti, Daniel C Silva, and Eugénio C Oliveira. 2018. BIM, GIS, IoT, and AR/VR integration for smart maintenance and management of road networks: a review. In *2018 IEEE international smart cities conference (ISC2)*. IEEE, 1–7.

[5] Christina Chaccour, Mehdi Naderi Soorki, Walid Saad, Mehdi Bennis, and Petar Popovski. 2022. Can terahertz provide high-rate reliable low latency communications for wireless VR? *IEEE Internet of Things Journal* (2022).

[6] David Goedicke, Alexandra WD Bremers, Hiroshi Yasuda, and Wendy Ju. 2021. Xr-oom: Mixing virtual driving simulation with real cars and environments safely. In *13th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*. 67–70.

[7] Sandra Helsel. 1992. Virtual reality and education. *Educational Technology* 32, 5 (1992), 38–42.

[8] Min Huang and Xu Zhang. 2018. MAC scheduling for multiuser wireless virtual reality in 5G MIMO-OFDM systems. In *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 1–6.

[9] Ran Ju, Jun He, Fengxin Sun, Jin Li, Feng Li, Jirong Zhu, and Lei Han. 2017. Ultra wide view based panoramic VR streaming. In *Proceedings of the Workshop on Virtual Reality and Augmented Reality Network*. 19–23.

[10] Zeqi Lai, Y Charlie Hu, Yong Cui, Linhui Sun, Ningwei Dai, and Hung-Sheng Lee. 2019. Furion: Engineering high-quality immersive virtual reality on today's mobile devices. *IEEE Transactions on Mobile Computing* 19, 7 (2019), 1586–1602.

[11] Ajey Lele. 2013. Virtual reality and its military utility. *Journal of Ambient Intelligence and Humanized Computing* 4, 1 (2013), 17–26.

[12] Luyang Liu, Ruiguang Zhong, Wuyang Zhang, Yunxin Liu, Jiansong Zhang, Lintao Zhang, and Marco Gruteser. 2018. Cutting the cord: Designing a high-quality untethered vr system with low latency remote rendering. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*. 68–80.

[13] Zhihan Lv, Tengfei Yin, Xiaolei Zhang, Houbing Song, and Ge Chen. 2016. Virtual reality smart city based on WebVRGIS. *IEEE Internet of Things Journal* 3, 6 (2016), 1015–1024.

[14] Simone Mangiante, Guenter Klas, Amit Navon, Zhuang GuanHua, Ju Ran, and Marco Dias Silva. 2017. Vr is on the edge: How to deliver 360 videos in mobile networks. In *Proceedings of the Workshop on Virtual Reality and Augmented Reality Network*. 30–35.

[15] Wendy Powell, Tom Alexander Garner, Seth Shapiro, and Bryce Paul. 2017. Virtual reality in entertainment: The state of the industry. (2017).

[16] Feng Qian, Lusheng Ji, Bo Han, and Vijay Gopalakrishnan. 2016. Optimizing 360 video delivery over cellular networks. In *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*. 1–6.

[17] Dipankar Raychaudhuri, Ivan Seskar, Gil Zussman, Thanasis Korakis, Dan Kilper, Tingjun Chen, Jakub Kolodziejski, Michael Sherman, Zoran Kostic, Xiaoxiong Gu, et al. 2020. Challenge: COSMOS: A city-scale programmable testbed for experimentation with advanced wireless. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*. 1–13.

[18] Greg S Ruthenbeck and Karen J Reynolds. 2015. Virtual reality for medical training: the state-of-the-art. *Journal of Simulation* 9, 1 (2015), 16–26.

[19] José Santos, Tim Wauters, Bruno Volckaert, and Filip De Turck. 2017. Fog computing: Enabling the management and orchestration of smart city applications in 5G networks. *Entropy* 20, 1 (2017), 4.

[20] Serkan Solmaz, Jessica L Dominguez Alfaro, Pedro Santos, Peter Van Puyvelde, and Tom Van Gerven. 2021. A practical development of engineering simulation-assisted educational AR environments. *Education for Chemical Engineers* 35 (2021), 81–93.

[21] Marko Viitanen, Jarno Vanne, Timo D Hämäläinen, and Ari Kulmala. 2018. Low latency edge rendering scheme for interactive 360 degree virtual reality gaming. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 1557–1560.

[22] Zhehui Zhang, Shu Shi, Varun Gupta, and Rittwik Jana. 2019. Analysis of cellular network latency for edge-based remote rendering streaming applications. In *Proceedings of the ACM SIGCOMM 2019 Workshop on Networking for Emerging Applications and Technologies*. 8–14.