Can My WiFi Handle the Metaverse? A Performance Evaluation Of Meta's Flagship Virtual Reality Hardware

Matthijs Jansen*
Jesse Donkervliet*
Vrije Universiteit Amsterdam
The Netherlands
{M.S.Jansen,J.J.R.Donkervliet}@vu.nl

Animesh Trivedi
Vrije Universiteit Amsterdam
The Netherlands
A.Trivedi@vu.nl

Alexandru Iosup
Vrije Universiteit Amsterdam
The Netherlands
A.Iosup@vu.nl

ABSTRACT

Extending human societies into virtual space through the construction of a metaverse has been a long-term challenge in both industry and academia. Achieving this challenge is now closer than ever due to advances in computer systems, facilitating large-scale online platforms such as Minecraft and Roblox that fulfill an increasing number of societal needs, and extended reality (XR) hardware, which provides users with state-of-the-art immersive experiences. For a metaverse to succeed, we argue that all involved systems must provide consistently good performance. However, there is a lack of knowledge on the performance characteristics of extended reality devices. In this paper, we address this gap and focus on extended- and virtual-reality hardware. We synthesize a user-centered system model that models common deployments of XR hardware and their trade-offs. Based on this model, we design and conduct real-world experiments with Meta's flagship virtual reality device, the Quest Pro. We highlight two surprising results from our findings which show that (i) under our workload, the battery drains 15% faster when using wireless offloading compared to local execution, and (ii) the outdated 2.4 GHz WiFi4 gives surprisingly good performance, with 99% of samples achieving a frame rate of at least 65 Hz, compared to the 72 Hz performance target. Our experimental setup and data are available at https://github.com/atlarge-research/measuring-the-metaverse.

CCS CONCEPTS

• **Human-centered computing** → **Virtual reality**; • **Networks** → **Network measurement**.

KEYWORDS

metaverse, virtual reality, performance analysis, task offloading, virtual worlds, gaming, networking, computation

ACM Reference Format:
Matthijs Jansen, Jesse Donkervliet, Animesh Trivedi, and Alexandru Iosup. 2023. Can My WiFi Handle the Metaverse? A Performance Evaluation Of Meta's Flagship Virtual Reality Hardware. In *Companion of the 2023 ACM/SPEC International Conference on Performance Engineering (ICPE '23 Companion), April 15–19, 2023, Coimbra, Portugal.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3578245.3585022

*Both authors contributed equally to this research.

This work is licensed under a Creative Commons Attribution-ShareAlike International 4.0 License.

ICPE '23 Companion, April 15–19, 2023, Coimbra, Portugal
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0072-9/23/04.
https://doi.org/10.1145/3578245.3585022

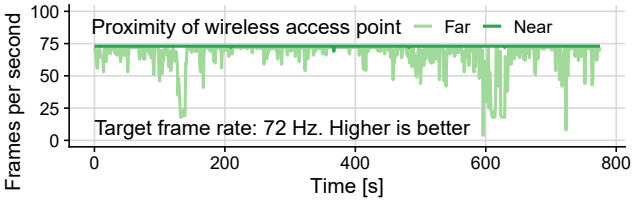


Figure 1: An example result from our performance characterization, showing the effect of wireless access point proximity on VR performance.

1 INTRODUCTION

Although the term *metaverse* was introduced in 1992 [15], a digital ecosystem that extends human society into the virtual space, its realization has shown to be challenging. Recent technological developments in hardware and software have caused a resurgence of interest in the topic, with Meta's investment of over $36 billion as the most prominent example. Today, closest to realizing the vision of a metaverse, online platforms such as Minecraft and Roblox have more than 100 million active users [3] and are increasingly used for activities beneficial to society, such as education [2], professional training [12], and social activism [14].

Among the technical challenges towards realizing a metaverse, novel user-interfacing hardware is required to provide users with an immersive experience. These devices are collectively known as *extended reality (XR)* devices, which is an umbrella term for virtual reality (VR), mixed reality (MR), and augmented reality (AR) [6], each of which extends the users perception of the physical world with virtual elements. Although a metaverse would impose strict performance and quality of service (QoS) requirements on such hardware, little is known about their performance. Existing studies primarily focus on the visual fidelity of devices but do not address the performance characteristics of metaverse systems and their components. Key trade-offs for these devices include the use of more powerful XR hardware to run more advanced applications, which increases the weight and cost of the headset, and decreases user-friendliness and usability. As a potential solution, more lightweight devices can offload work to nearby edge or cloud devices with sufficient processing power. However, this puts additional strain on local networks, which might not be feasible.

We argue that knowledge on performance behavior for XR hardware is vital for users and hardware and software developers to make more informed decisions when deploying or designing extended reality hardware. Figure 1 shows an example result that exemplifies this observation. The figure shows that the performance

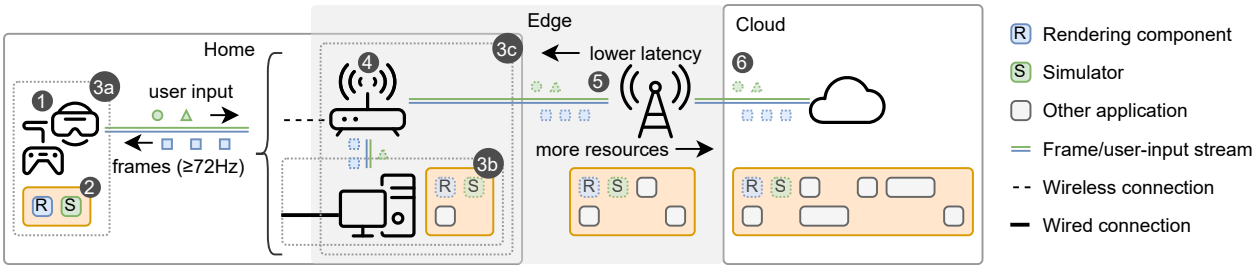


Figure 2: User-centered eXtended Reality (XR) system model.

delivered by a state-of-the-art XR device can vary significantly, from good performance to rendering the device unusable, as a result of the proximity of the XR device to a wireless access point. *In general, we observe a general lack of knowledge on how XR hardware can be deployed to partake in the metaverse and what trade-offs in performance, user-friendliness, and cost exist for these deployments.*

In this work, we address this knowledge gap and propose a user-centered system model for an extended-reality metaverse that describes the components involved in participating in the metaverse using extended-reality hardware and hosting services in the metaverse. We define the performance characteristics and interactions of these components to present a clear overview of available metaverse deployments and research opportunities for improving how users interact with the metaverse. These deployments include native processing on XR headsets, as well as wired and wireless task offloading to edge or cloud devices, and present various trade-offs in resource utilization, QoS, and energy efficiency. To quantify these trade-offs, we design and conduct real-world experiments using the Meta Quest Pro, Meta's flagship virtual reality device. Based on the experiment results, we provide a performance analysis of virtual reality workloads. We further analyze metrics related to network, processing power, and energy usage, and compare various deployment scenarios to provide actionable insights on how users can best participate in the metaverse, as well as how the metaverse should be developed in the future.

Our key contributions are:

(1) We synthesize a user-centered system model for extended reality use cases, defining common components in XR deployments. We analyze performance characteristics and trade-offs of each component and present gaps between current XR deployments and possible deployments, showing research opportunities for improving the metaverse (Section 2).
(2) We translate the abstract components from our system model into concrete user concerns, classify key metrics for answering these concerns, and synthesize a list of experiments for capturing these metrics under various workloads and deployments related to headset resource usage, VR task offloading, and required network quality (Section 3).
(3) We perform our experiments on virtual reality deployments for native processing and wired and wireless offloading to the edge. We capture application- and system-level metrics to quantify the performance differences between deployments, and present actionable insights for users and developers to increase QoS (Section 4).

2 SYSTEM MODEL

Figure 2 shows our user-centric model for XR applications. To access XR applications, users put on a headset or glasses that include one or several displays (component ❶ in Figure 2). For VR applications, the headset blocks outside light, and the displays cover the user's eyes. For MR and AR applications, either glasses or VR headsets with live video pass-through are used, mixing physical and virtual objects on the same screen.

The main responsibilities of the XR system are (i) to relay the user's position, orientation, and other input to a simulator, and (ii) to obtain rendered frames and show them to the user, creating a visual representation of the virtual world or objects. The device meets these responsibilities by communicating with a virtual world simulator (S) and rendering component (R) respectively (❷). To provide good quality of experience (QoE), the device must respond to the user's input with low latency. Therefore, user input sampling and output frame rate must run at a consistently high rate. In practice, a common minimum refresh rate is 72 Hz (i.e., 14 ms frame time), which is well below the update lag threshold of 48 ms that is known to cause motion sickness [5].

In general, the simulator and rendering component are part of a data processing pipeline whose components can be deployed in different environments. For example, for online multiplayer games, the simulator commonly runs on a cloud-based server, while the rendering component runs on the user's device. In Figure 2, orange boxes indicate environments where the rendering component and simulator can be deployed. Choosing where to deploy these components implies navigating a trade-off between latency and available resources. Deployment on or near the user's device guarantees low latency and limited resources, which results in good QoE as long as no resource-intensive simulation or graphics rendering is required. Our model incorporates a cloud-edge continuum [8], in which environments further from the user offer increasingly more resources, at the expense of increasingly higher latency.

Although the simulator and rendering component can be deployed at different locations, current practice commonly places these two components together because of their interdependency. Our model shows three deployments common in today's commercial XR systems. First, devices simulate and render on the user's device (3a). This allows the device to operate wirelessly, increasing the mobility of the user. However, the device must provide sufficiently powerful hardware to simulate and render the virtual environment or objects, and be equipped with a battery that can

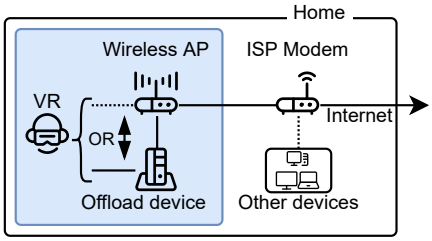


Figure 3: Experimental Setup. Experiments focus on the area in blue. Solid and dotted lines represent wired and wireless connections respectively.

power the device for at least several hours. Popular devices using this configuration are the Meta Quest 2 and Quest Pro.

Second, devices can offload the simulation and rendering to other nearby devices (3b). In this case, the device is typically directly connected to the device using a USB 3.0 connection. This allows reusing the (more powerful) resources from another user device and maintains low latency response times. However, the wired connection limits the user's ability to navigate their physical space while using the device, and the user must own or buy hardware that is sufficiently powerful to support the offloaded workload, which can exceed the price of the XR device. Well-known devices supporting this configuration are the Meta Quest 2, Quest Pro, PlayStation VR2, and Valve Index.

Third, and finally, devices can support the combination of the previous two approaches. Here, simulation and rendering are wirelessly offloaded to another device with more powerful hardware (3c). This approach provides both high user mobility and an opportunity for computationally intensive simulation and rendering. However, this approach requires support from the user's wireless local area network (WLAN, 4), which must consistently provide sufficient bandwidth to stream video at at least 72 frames per second. As such, it is important for the user to answer the following questions: *what type and quality of network does the XR headset require? How do I connect the headset to the internet? Can the headset process XR workloads on its own, or do I have to offload, and, if so, where to?*

We briefly consider here the research challenges and opportunities of connecting XR devices to WLAN, wide area networks (WANs), and the internet, and argue that it creates two additional important opportunities. First, it allows offloading to local edge devices (WLAN at 3b) and remote edge and cloud environments (5 and 6), increasing the user's choice in trading off latency for additional resources. Offloading to edge and cloud datacenters for extended reality workloads is an active topic of research and is discussed in Section 5. Second, it allows the user to participate in *online applications*, enabling various additional use cases ranging from virtual tourism to remote surgery. As such, important questions for internet service providers (ISPs) are *what are the performance and latency requirements from users for varying types of applications? What are the bandwidth requirements needed to support these applications? And how many concurrent users must the platform support?* Exploring answers to these questions is part of ongoing work. In the remainder of this work, we focus on the WLAN environment.

3 EXPERIMENT DESIGN

In our system model, we showed that it is interesting for users to understand how the WLAN affects the performance, and specifically the frame rate, of XR devices, and how different deployments affect the device's battery life. In this section, we attempt to answer these questions through experiments using the VR headset Meta Quest Pro, and provide actionable recommendations for users to setup or improve their VR deployment, and for edge and cloud service providers to better support VR workloads. We leave the analysis of AR and MR hardware to future work. For our experimental design, we incorporate the various discussed VR deployments such as native processing and wired and wireless offloading, and translate these abstract descriptions into concrete experimental setups.

We attempt to answer the following fundamental questions:

- **Q1 What is the performance and resource usage of VR applications on state-of-the-art VR hardware?** We benchmark the VR game Beat Saber on Meta's flagship virtual reality hardware the Meta Quest Pro, and demonstrate that its resources are powerful enough to deliver a high-quality and stable user experience, but that resource utilization is highly skewed between CPU, GPU, and memory (Section 4.1).
- **Q2 What are the advantages and disadvantages of VR workload offloading compared to native processing on VR headsets?** We offload a VR workload to a nearby desktop PC using wired and wireless connectivity, and show how both scenarios result in flawless user experience, while drastically changing the headset's resource usage (Section 4.2).
- **Q3 What are the network requirements to enable wireless compute offloading for VR?** We perform an in-depth comparison of wireless compute offloading for VR and vary WiFi signal strength (Section 4.3) and WiFi configurations (Section 4.4) to show how seemingly small changes in VR setups have a significant effect on performance.

For our experiments, we explore virtual reality deployments in the WLAN between VR headsets and edge devices using wired and wireless networks. We demonstrate our experimental setup in Figure 3, which is one instantiation of our abstract system model. We leave a broader exploration of our model to future work. We use the flagship virtual reality headset Meta Quest Pro, which uses a Qualcomm Snapdragon XR2+ CPU, Adreno 650 GPU, and 12 GB memory. Its screens have a resolution of 1800×1920 pixels each and support refresh rates of 72 and 90 Hz. We use a desktop computer as a local edge device with an Intel i5-12400F CPU, an Nvidia RTX 3080 GPU, and 32 GB memory. We connect the headset and desktop computer via a USB-C 3.0 link cable or a TP-Link Archer AXE75 WiFi6E access point. Unless otherwise indicated, we configure the access point to use WiFi6 at 5 GHz (i.e., a/n/ac/ax), and use the Meta Quest Pro within five meters distance with clear line of sight.

For our experiments, we use the gaming applications Beat Saber and Half-Life: Alyx. Gaming applications represent a significant portion of the total VR market [7], and these games are two representative examples in this space. Beat Saber is a popular cross-platform application and one of the best-selling VR games, with over 4 million copies sold. Half-Life Alyx is a highly popular game and was nominated for 4 BAFTA game awards in 2021.

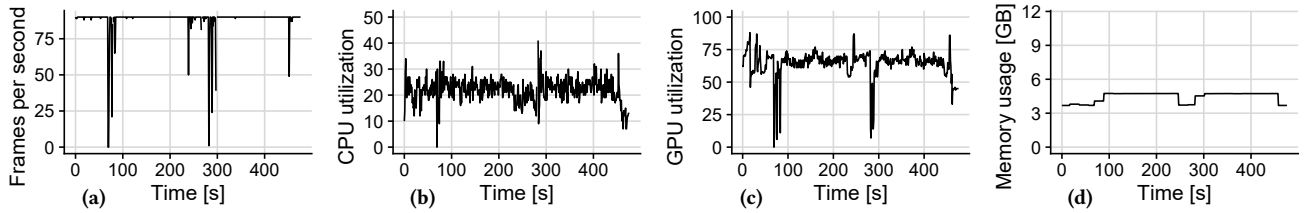


Figure 4: Resource usage of VR applications natively running on the Meta Quest Pro while playing the game Beat Saber.

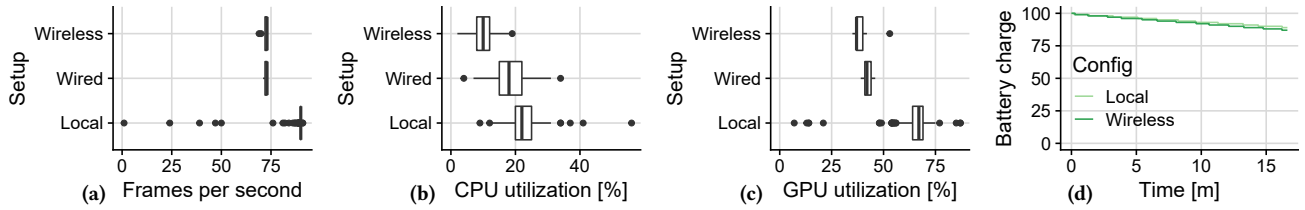


Figure 5: Performance and resource usage comparison for three device setups. Local indicates rendering video on the Meta Quest Pro. Wired and Wireless indicates streaming video rendered on a PC over a wired and wireless connection respectively.

Although our experiments use one XR device and two gaming applications, we expect our results to generalize beyond this scope. Specifically, when offloading the gaming application to an edge device, the VR device's main task is to decode the incoming video stream. We expect the performance of this task to scale with known hardware and video quality properties. When running applications on the VR device itself, we expect performance differences between applications, depending on their computational demands, and over time, as developers increasingly find ways to efficiently use the available hardware resources.

When offloading, we use the game platform Steam to execute applications on the desktop computer as the Oculus App, which facilitates communication between the headset and the PC, can not run games itself. The Meta Quest Pro uses Android, which allows us to collect metrics from the headset using ADB and Logcat on the desktop computer. We use the Python library psutil to collect system-level metrics from the desktop computer. All metrics are sampled once per second. We capture network activity from the Meta Quest Pro using Wireshark on the PC. The headset and laptop are the only two devices wirelessly connected to the access point.

4 EXPERIMENTS

In this section, we analyze the results obtained from our experiments and discuss our main findings. We discuss Experiment **Q1** and **Q2** in Sections 4.1 and 4.2 respectively. We split our findings from Experiment **Q3** into two parts, and discuss these in Sections 4.3 and 4.4. Our main findings are summarized at the end of each of these sections, respectively.

4.1 Virtual Reality Performance Analysis

For our first experiment, we examine the Meta Quest Pro performance and resource utilization when executing VR applications to *establish a baseline of local processing for later experiments on task offloading to the edge.* We use the game Beat Saber as workload, where users actively move their headset and controllers, and report metrics from 8 minutes of play time in Figure 4. The plots show the game's performance over time, measured in frames per second, and resource utilization of the headset's CPU, GPU, and memory.

Figure 4a shows that the game achieves stable performance of 90 Hz, the headset's maximum frame rate, with the exception of some steep drops during the game's load screens, which do not hinder the user experience. Figures 4b-4d show that the GPU's utilization is significantly higher than the other resources, indicating a possible performance bottleneck for applications with higher-quality graphics. In comparison, CPU utilization is relatively low: between 15% and 32% while playing. Outliers in CPU and GPU usage coincide with loading screens, and do not affect user experience.

Finally, Figure 4d shows the memory usage during gameplay, with 12 Gb being the maximum memory capacity of the Meta Quest Pro. The plot shows stable memory usage between 3 and 5 GB, only increasing and decreasing when a level is loaded. This indicates that, for this game, the complete level is loaded into memory before the game starts. This is feasible because the level has a fixed size and duration, which means dynamically loading data into memory can be avoided. However, this behavior may not be possible for other types of (gaming) applications with larger memory footprints. For these applications, it is necessary to load data on demand from the headset's local storage or over the internet.

Main Finding 1: We deploy a gaming use case on the Meta Quest Pro, and show that GPU utilization is significantly higher than that of the CPU and memory. This insight helps users understand what applications can be played on their headset with optimal QoE, and what applications require a hardware upgrade or offloading. For hardware vendors, these insights help them prioritize hardware component upgrades for the next generation of XR headsets to better support user application requirements.

4.2 Compute Offloading to the Edge

For our second experiment, we offload the game Beat Saber to a nearby desktop PC to answer the question *if offloading VR applications to nearby edge devices is feasible, and if so, what offloading deployments can be used to achieve this.* The desktop PC runs the application and streams the video to the headset using wired or wireless networks. We use a USB-C 3.0 cable for our wired offloading deployment, and WiFi for wireless offloading. Each configuration runs once and takes between 7 and 8 minutes to complete. We

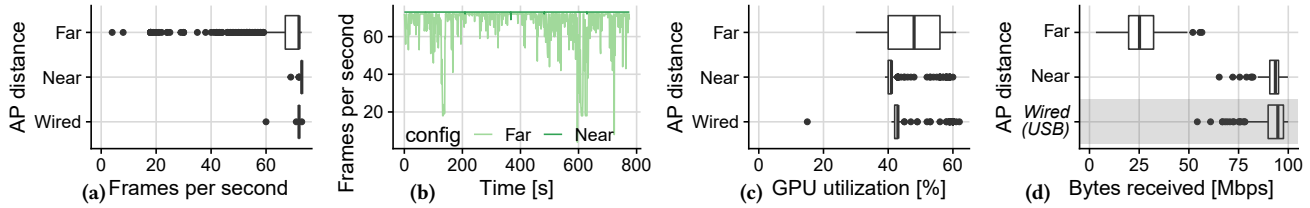


Figure 6: Performance and resource usage comparison for three device setups. The Near and Far configurations use a wireless connection, nearby and far from the wireless access point respectively.

present our offloading results in Figure 5. We exclude time spent on putting on the headset and starting the application, and plot 225 seconds of data for Figures 5a to 5c. To compare battery consumption, shown in Figure 5d, we perform an extended run of more than 15 minutes and plot all collected data. We compare our results against our baseline (shown in Figure 4).

Our results in Figure 5a show that all three deployments provide good performance, measured in frames per second, but significantly differ in CPU and GPU usage (Figures 5b-5d). We omit memory usage as this is constant over time as previously shown in Figure 4. Local processing does achieve a higher frame rate of 90 Hz compared to 72 Hz when offloading: analysis shows that this is caused by the configuration of the components in our experiment setup, and does not indicate that local applications perform better. Surprisingly, the wireless and wired setups perform almost identically, providing evidence that wireless video streaming to VR devices with good performance is possible. For this reason, we analyze wireless offloading in-depth in Sections 4.3 and 4.4.

A second insight is that the native deployment is the only deployment with significant performance variation. This is expected because the headset can lower the frame rate during (static) loading screens without the user noticing. This is not taken into account when offloading, however, with the offloaded application maintaining a stable frame rate under all conditions.

Figure 5b shows the CPU utilization for each setup. Native processing uses the most CPU resources as it has to execute the game itself. More surprisingly, wireless offloading uses significantly less CPU resources than wired offloading. The median CPU usage for wireless is approximately 10%, while the median for the wired setup is close to 20%. Intuitively, the CPU utilization for wired and wireless offloading should be similar as both offload the game to the edge. We hypothesize that the difference in CPU usage is caused by the difference in how hardware and software handle USB and WiFi network traffic. A dedicated network card in the Meta Quest Pro can assist the CPU in processing the incoming wireless data stream; for wired USB offloading such dedicated hardware might not be present. Overall, the CPU utilization is low, with all samples, including outliers, indicating utilization below 60%.

In Figure 5c, we compare GPU utilization. Here, GPU usage is lower for both offloading scenarios than native processing as they offload graphics rendering responsibilities to the edge. However, the gap in GPU usage is only 25%, with wired and wireless offloading seeing a GPU usage between 40% and 45%. We argue that GPU usage when offloading is still high because the GPU needs to spend significant resources on decoding the video signal sent from the edge. Video encoding is not required when running the application on the headset, resulting in the small gap in GPU usage.

Finally, we measure the battery usage from the start of the play session, and report our results in Figure 5d. We exclude wired offloading in our comparison as this deployment is charged over the USB-C cable, keeping the headset's charge around 100%. Surprisingly, we see that the wireless offloading deployment (dark green in the plot) uses slightly more battery than the native deployment, although both CPU and GPU usage is lower when using wireless offloading. Extrapolating battery usage for both configurations shows expected battery life of 128 minutes and 152 minutes respectively. We hypothesize that the network card of the Meta Quest Pro is responsible for this unforeseen difference in battery usage, working hard on receiving and processing the incoming WiFi data stream. Headsets specialized for wireless offloading could improve battery life over native processing by using more energy-efficient hardware.

Main Finding 2: We offload a VR application to a local edge device over wired and wireless networks, and demonstrate that both offloading deployments achieve comparable performance to native processing, but significantly change resource usage and battery life. This information can be used by hardware vendors to create specialized headsets for offloading, and by users to predict what battery life their deployment can achieve.

4.3 Wireless Networking Requirements

For our third experiment, we evaluate offloading deployments with various wireless network qualities to answer the question of *what is the minimum required network quality to do VR offloading, and how does this affect performance?* More specifically, we vary the distance and number of objects between the VR headset and a wireless access point. We compare three deployments: (i) Far, where the user is located 10 m from the access point, with a wall in between; (ii) Near, where the user is 5 m from the access point with a clear line of sight, and (iii) Wired, a baseline that uses wired offloading. Each configuration runs once and is plotted using at least 775 seconds of data. We report here our findings for the application Half-Life: Alyx. We omit our results for Beat Saber, but observe similar network performance for both applications.

Our evaluation in Figures 6a and 6b shows a significant performance difference between the *near* and *far* offloading deployments, with *near* performing as good as the wired offloading baseline, and *far* achieving as low as 4 frames per second. This difference shows that although wireless offloading for VR video streaming is possible, it is only possible under good conditions. The inconsistent performance of the *far* deployment is caused by the high distance and

the number of objects between the headset and the access point. This weakens the strength of the WiFi signal, which lowers the available network bandwidth below the required bandwidth for offloading 72 frames per second (Figure 6d). The 25th percentile frames per second for the far deployment is still 67 Hz (Figure 6a), but the frequent frame drops that occur throughout the experiment (Figure 6b) can already cause motion sickness [5].

Surprisingly, the average GPU utilization is highest in the far deployment with a median utilization of 48% compared to a utilization of 41 and 43% for the near and wired deployments (Figure 6c). The far deployment was expected to have the lowest GPU utilization because a lower average frame rate should result in fewer frames to be rendered by the headset's GPU. We conjecture that the increase in GPU utilization is caused by the Meta Quest Pro's choice of streaming over TCP, causing dropped packets to be retransmitted, leading to large batches of frames to be delivered to the device simultaneously, once the dropped packet has been received.

Main Finding 3: We compare VR deployments with strong (90 Mbps) and weak (25 Mbps) WiFi signals, and find that although wireless offloading for VR workloads is possible, it requires stable, high throughput networks to maintain 72 frames per second.

4.4 Wireless Networking Settings

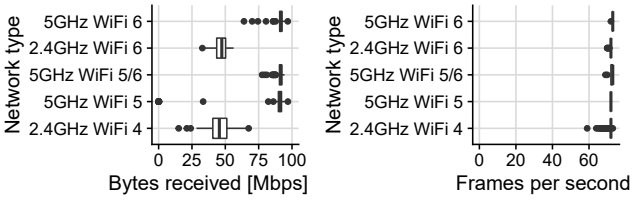
For our final experiment, we wirelessly offload the game Beat Saber to a nearby access point with a clear line of sight, and change various WiFi settings to further investigate *how WiFi quality affects VR offloading performance.* We switch in our experimental setup between WiFi standards 4, 5, and 6, and between network frequencies of 2.4 Ghz and 5 Ghz. We run each configuration once and report 200 seconds of data collected after the application has stabilized.

Figure 7a shows that the biggest discriminator for network throughput usage is the frequency of the wireless network. The median bandwidth usage for 2.4 Ghz networks is 50 Mbps, whereas this is 91 Mbps for 5 Ghz networks. The maximum bandwidth usage never exceeds 100 Mbps—the Meta Quest Pro's default maximum bandwidth. The resulting frame rate (Figure 7b) differs surprisingly little between the varying WiFi types. Even for 2.4 Ghz WiFi4, with the lowest median bandwidth of 48 Mbps, 99% of samples achieve above 65 Hz, compared to a targeted frame rate of 72 Ghz.

Main Finding 4: WiFi frequency has a significant effect on the available bandwidth between a VR headset and offloading target, but surprisingly little on the frame rate, with even the worst configuration achieving 65 frames per second in 99% of samples. However, this is achieved with a high-quality, uninterrupted signal between the headset and network access point; these results do not generalize to arbitrary access point placement. However, we consider a scenario where the WiFi signal has to traverse a wall in Figure 6.

5 RELATED WORK

To the best of our knowledge, we are the first to present an in-depth performance analysis of virtual reality workloads using physical VR hardware. However, related topics such as computer vision and hardware analysis for VR, task offloading, and simulation and performance modeling of VR workloads have been well studied. We highlight the most important related work in this section.



(a) Network bandwidth usage. **(b) Frame rate. Higher is better.**

Figure 7: Network usage and performance of the Meta Quest Pro while streaming a game close to a wireless access point. Bandwidth usage is limited to 100 Mbps.

Visual perception: The display in virtual reality headsets is positioned near the user's eyes, contrary to traditional 2D computer displays. With the short distance between the eyes and the screen in VR (and AR) headsets, displays with a high refresh rate are required to prevent motion sickness and give an overall good user experience. Finding the optimal refresh rate and head tracking latency for user experience has been a well-researched topic [10, 11, 16].

Task offloading: Running virtual reality workloads requires significant computing power which may not be available or desirable on an XR headset. Therefore, the offloading of XR tasks to nearby desktop computers is a popular deployment option, and has seen significant attention from the research community. Nyamtiga et al. have studied offloading VR tasks to edge infrastructure [13]. To this, we add work from Zhu et al. focused on power efficiency when offloading [17]. Alshahrani et al. have researched computation offloading to cloud and edge for multi-player applications [1]. Cozzolino et al. have focused on the offloading of AR applications on the other hand [4]. Finally, Korneev et al. have studied the relation between network conditions and user experience when offloading, and how changing the network conditions impacts the user experience [9]. However, their evaluation is limited to simulation.

6 CONCLUSION AND ONGOING WORK

Extending human societies into virtual space through the construction of a metaverse has been a long-term challenge in both industry and academia. However, recent developments in both hardware and software systems bring the creation of a successful metaverse closer than ever. An important part of this effort is providing users with an immersive experience through the use of extended reality (XR) devices. However, little is known about the performance of these devices, which must provide consistently good performance and meet strict latency requirements. Furthermore, a variety of different device and deployment designs exist, creating confusion for users and developers, and making it difficult to compare and understand the performance characteristics of these devices.

To this end, we present a system model for extended reality devices and use it to design and conduct real-world experiments. Our experiments evaluate Meta's flagship VR device, the Quest Pro, using three deployments commonly observed in practice, including task offloading to the edge. Based on our experiment results, we show that state-of-the-art consumer wireless access points can only maintain good performance under favorable conditions, and that VR devices can achieve good frame rates even when using older,

2.4 GHz, WiFi networks. We further show that the Meta Quest Pro's GPU has significantly higher utilization than other measured system components under regular use, and that streaming video to the Meta Quest Pro over WiFi drains the battery faster than native processing, regardless of the lower resource utilization.

This work is part of ongoing work that aims to provide a detailed understanding of the performance characteristics of XR devices for metaverse applications. To this end, we plan to perform further experiments evaluating device performance for a variety of workload types (other than games) and obtain a more detailed understanding of the device's battery usage and energy efficiency. Our experimental setup and data is available at https://github.com/atlarge-research/measuring-the-metaverse.

ACKNOWLEDGMENTS

This work is supported by NWO grant OffSense (OCENW.KLEIN.209), and by structural funds from VU Amsterdam.

REFERENCES

[1] Alshahrani et al. 2020. Efficient Multi-Player Computation Offloading for VR Edge-Cloud Computing Systems. *Applied Sciences* 10, 16 (2020).
[2] Bar-El and Ringland. 2020. Crafting Game-Based Learning: An Analysis of Lessons for Minecraft Education Edition. In *FDG '20*.
[3] Boddy. 2021. Minecraft boasts over 141 million monthly active users and other impressive numbers. *Windows Central* (2021).
[4] Cozzolino et al. 2022. Nimbus: Towards Latency-Energy Efficient Task Offloading for AR Services. *IEEE Transactions on Cloud Computing* (2022).
[5] Draper. 1998. *The adaptive effects of virtual interfaces: vestibulo-ocular reflex and simulator sickness.* Ph. D. Dissertation.
[6] Farshid et al. 2018. Go boldly!: Explore augmented reality (AR), virtual reality (VR), and mixed reality (MR) for business. *Business Horizons* 61, 5 (2018).
[7] Fortune Business Insights. 2021. Virtual Reality (VR) in Gaming Market Size, Share & COVID-19 Impact Analysis, By Component, By Device, and Regional Forecast, 2021-2028. https://www.fortunebusinessinsights.com/industry-reports/virtual-reality-gaming-market-100271.
[8] Jansen et al. 2022. The SPEC-RG Reference Architecture for the Edge Continuum. https://doi.org/10.48550/ARXIV.2207.04159
[9] Korneev et al. 2022. Studying Cloud-Based Virtual Reality Traffic. *Journal of Communications Technology and Electronics* 67, 12 (2022).
[10] Livingston et al. 2013. *Basic Perception in Head-Worn Augmented Reality Displays.* Springer New York.
[11] Mania et al. 2004. Perceptual sensitivity to head tracking latency in virtual environments with varying degrees of scene complexity. In *APGV 2004*, Vol. 73.
[12] Michele Melchiorre, Senior VP BMW. 2022. The BMW iFactory - Highly Efficient and Competitive. In *Keynote ISC*.
[13] Nyamtiga et al. 2022. Edge-Computing-Assisted Virtual Reality Computation Offloading: An Empirical Study. *IEEE Access* 10 (2022).
[14] Reporters without Borders. [n. d.]. *The Uncensored Library.* https://www.uncensoredlibrary.com/
[15] Stephenson. 1992. *Snow Crash.* Bantam Spectra Books.
[16] Zhao et al. 2017. Estimating the motion-to-photon latency in head mounted displays. In *IEEE VR 2017*.
[17] Zhu et al. 2022. Power-Efficient Live Virtual Reality Streaming Using Edge Offloading. In *NOSSDAV*.