Performance Evaluation of a Multi-User Virtual Reality Platform

Venkatakrishnan Parthasarathy*, Anderson Augusto Simiscuka†, Noel O'Connor‡ and Gabriel-Miro Muntean§
School of Electronic Engineering, Dublin City University, Dublin
E-mails: *venkatakrishnan.parthasarathy2@mail.dcu.ie, †anderson.simiscuka2@mail.dcu.ie, ‡noel.oconnor@dcu.ie, §gabriel.muntean@dcu.ie

***Abstract*—Virtual Reality (VR) popularity is increasing as it is becoming more affordable for end users. Available VR hardware includes low-end inexpensive devices like Google Cardboard and high-end ones like HTC Vive or Oculus Rift, which are more expensive headsets. Using VR as a platform for content delivery allows better user engagement than other traditional methods, as VR headsets remove external distractions. Multi-user VR applications provide shared experiences where users can communicate and interact in the same virtual space. This shared environment, however, introduces challenges regarding network performance, quality of service (QoS) and sessions privacy. This paper presents a multi-user VR application and aims to evaluate network behaviour in a number of scenarios, including real VR headsets (i.e. Oculus Rift), as well as simulated ones. This QoS analysis is important for the understanding of how many VR users can be simultaneously connected with high image quality.**

***Index Terms*—Virtual Reality, Multi-User, Network Performance, Quality of Service.**

I. Introduction

Virtual Reality (VR) environments are the combination of 3D spaces with a virtual representation of a user [1]. VR users interact with polygonal 3D objects, created with 3D computer graphics software like Blender [2]. VR applications primary goal is to create a very immersive experience.

Many VR applications are focused on single-user experiences, such as educational and medical applications, physiotherapy software, prototyping programs for the car industry, museum tours, etc. [3], [4], [5]. Multi-user VR applications are designed for more than one user to be present in the virtual space at the same time. There are various parameters to be considered when developing such an application, as multi-user applications introduce challenges related to network performance, synchronisation among users and security to the system [6], [7], [8], [9].

Regarding connectivity in multi-user VR applications, traditional game server architecture can be employed. This architecture normally consists of a session server, which would have a list of all the users in a database. A game server is responsible for creating instances of the VR game and providing access to the users in the session server [10].

Based on these concepts, this paper introduces a VR application where multiple remote users interact in a virtual space, using the Oculus Rift. Fig.1 illustrates the application, which allows users to use the tablet component in the virtual room to select multimedia content to be played in the white screen. In addition, users can draw and communicate in a shared whiteboard available in the virtual environment, and also share files within the application. The application also allows simulated users to be added into the virtual space, allowing several scenarios with different numbers of users to be analysed. This quality of service (QoS) analysis is important for the understanding of how many VR users can be simultaneously connected with high image quality.

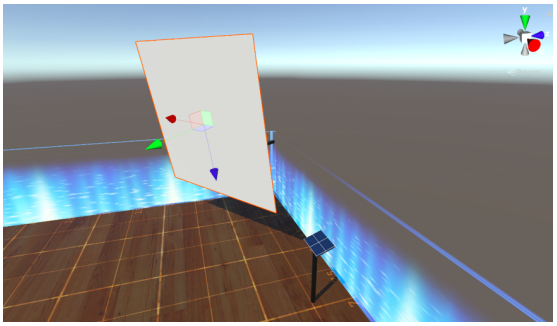

Fig. 1. VR environment with video player

This paper is organised as follows. In section II, related works are presented and section III details the design and implementation of the testbed. Section IV presents testing and results. Section V finalises the paper indicating the conclusions and future work directions.

II. Related Works

A. VR Development

Current VR devices are head-mounted [11] and display the virtual environment to the users. Some headsets contain audio outputs, which might have a surround sound audio delivery system, like the Oculus Rift [12]. Oculus Rift and HTC Vive [13] are two of the major VR hardware manufacturers. Windows Mixed Reality is another major player in the industry with multiple manufacturers producing VR hardware under the same software platform. There are also platform specific VR hardware devices like PSVR which is specifically designed for Sony's PlayStation platform [14]. Smartphones can also be used as VR devices when inserted into VR headsets (e.g. Google Cardboard) [15].

In order to support software development for VR, many software manufacturers are working on the development of Software Development Kits (SDKs), as presented in Table I.

OpenVR [16] is a hardware independent VR runtime and API developed by Valve Corporation while SteamVR is an OpenVR-based SDK that supports multiple platforms [17].

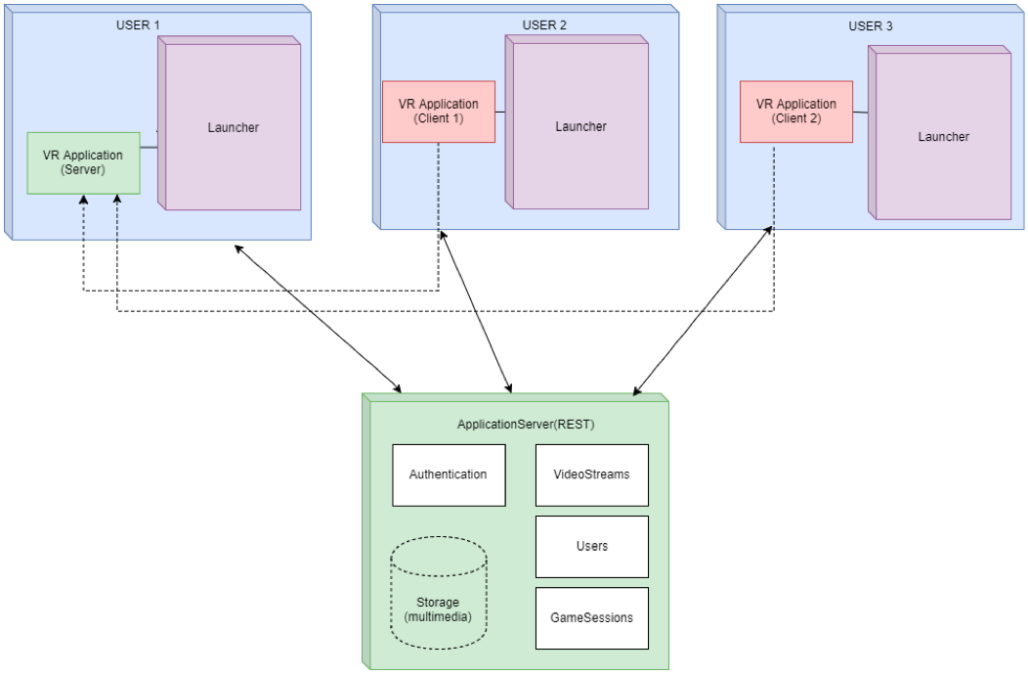

Fig. 2. Solution architecture

TABLE I
SUPPORT OF SOFTWARE DEVELOPMENT KITS BY DIFFERENT VR PLATFORMS

Software Development Kits	**Oculus Rift**	**HTC Vive**	**Windows Mixed Reality**	**Google Card-board**
OpenVR	Yes	Yes	No	No
SteamVR	Yes	Yes	Yes	No
MRTK	No	No	Yes	No
VRTK	Yes	Yes	Yes	Yes
Google VR SDK	No	No	No	Yes

MRTK or Mixed Reality Toolkit [18] was developed by Microsoft for developing VR applications for its Windows Mixed Reality Headsets. VRTK (Virtual Reality ToolKit) [19] is a tool that allows development of VR applications in an integrated environment with support to OpenVR, SteamVR, Unity and others. Google has its own SDKs [20] for Android and iOS Cardboard applications and the Daydream platform.

Multiple Game Development Environments (GDE) are currently available for developers. Deciding on which GDE should be using when building an application depends on the supported headsets, documentation, licensing, tools, compatibility with other components such as the computer graphics formats, etc. The GDE considered for the application presented in this paper is Unity [21], which fully supports the Oculus Rift. Unity also allows the game instance to run in a headless mode, where the graphic assets are not loaded in the server. This reduces the memory footprint on the server and allows a greater number of instances to run on the same computer.

B. Multi-User VR Application Design

Privacy is an important aspect to be considered in a multi-user application. Privacy can refer to private sessions, as well as the access to private information. Private virtual rooms can be created by the use of sessions and authorisation. In terms of privacy regarding personal information, such as user location, encryption is required and must not be accessible to the other users. All the non-sensitive information like player movement data and drawings/text in the shared whiteboard are sent to the clients in the system. User privacy is employed in the game session management and authentication of users is handled in game server instances. Users are authenticated with OAuth 2.0, an authentication protocol that retrieves login details from platforms such as Facebook and Twitter, registering users with their shared public details [22].

User interfaces in the application employ Electron, which uses standard web elements and can also be compiled across multiple platforms. The user interface contains the list of users and options to select the users who can access the game sessions [23].

The performance evaluation of network QoS parameters such as throughput and round-trip time (RTT) is needed to determine the robustness and usability of the application. Other non-network parameters such as the number of frames per second can be evaluated to check if the minimum requirements of the application are met [24], [25].

The work presented in this paper evolves the state-of-the-art presented in this section by introducing multiple simulated users to a Unity-made VR application. This paper also presents an analysis of the effects of the increased number of users on network performance, considering throughput, tick rate and RTT as the metrics for the evaluation of the multi-user approach. The works presented in this section focused either on privacy or network performance, while this paper evaluate the novel VR user-simulation algorithm in an application that considers both privacy and multiple scenarios with different number of users and network parameters.

III. Design and Implementation

The solution architecture is illustrated in Fig. 2. Design and implementation details are presented in this section. The implementation of the application allows for the testing of the proposed architecture and performance analysis of the solution.

The overall solution of this project consists of three components: application server, client launcher and the VR Application itself. The architecture presented in Fig. 2 indicates the interaction between the components in the solution. Each user's computer must have the VR application and the client launcher installed in their machines, which is also connected to the VR headset. The client launcher starts the VR application in server or client modes and creates game sessions in the cloud. It interacts with the application server to authenticate, authorise, create game sessions, manage users and measure network parameters for the appropriate server settings optimised for different network conditions.

The application server is a separate REST service intended to run in the cloud. It stores and serves multimedia data to the virtual screens accessible from the video stream resource endpoint.

A. Application Server

The application server is designed to be a cloud-based standalone server that facilitates authentication, access management, video streaming and game session management using RESTful endpoints. The application server was developed in Python and REST APIs. The states of the resources are stored in-memory to provide faster response.

The following endpoints are available:

1. All_users: A GET request on this endpoint returns all the users available on the system. It is used by launcher application to get the list of available users to select when creating a new game session in the GUI.

2. User: This endpoint is used to create, update and delete a user on the system.

3. Authentication: This endpoint takes a POST request with username and password as payload and authenticates users.

4. GameSession: This endpoint is used to create a game session with an IP address, owner and allowed users, storing this data in its payload. It is also used to retrieve server details for joining clients, with a GET request.

5. Video Stream: This endpoint allows the creation of new video stream links by uploading a new presentation or video file using a POST request. Presentation files such as .ppt and .pptx are converted into a slideshow video, and a video streaming link is generated. A GET request on this resource with the game session ID on the URL returns an object that contains the link to the video stream that will be displayed in the virtual environment.

B. Client Launcher

The client launcher is a GUI application written in Python. There are various screens in this component, developed for different use cases.

TABLE II
Network Status – Tick Rate Table

Network Status	Time Taken for First Connection	Tick Rate
1	>30 seconds	15 Hz
2	10 - 30 seconds	30 Hz
3	4 - 10 seconds	60 Hz
4	1 - 4 seconds	80 Hz
5	<1 seconds	120 Hz

All use cases of the launcher start with authenticating or creating a user. The screen consists of a form that requires a username and a password. After successful login, users can choose to create a new game session or to join an existing one.

After a new game session is created, a new game session ID is assigned to it, and the launcher performs a network analysis on with the VR application server. Based on the network speed and time taken for connection, a network status code is assigned. A tick rate is chosen based on the network status code, as seen in Table II. For instance, if the client took between 1 to 4 seconds to establish a connection with the server, the network status 4 will be assigned to the client. Each network status is related to the tick rate used in the game session. In the given example, the network status 4 (for an initial connection of 1-4s) represents a tick rate of 80Hz, which is suitable for this network, given the initial delay for the client to reach the server.

The tick rate is the number of times the states in the game synchronise between server and clients. The VR application is then launched with the newly applied tick rate. On a hosted server instance, the tick rate is the same as the number of frames per second (FPS). Therefore, the FPS will be higher for clients that can establish a connection with the server quickly.

Once the launcher displays all the details regarding the newly created game session, the game session ID can be shared with other users. The VR application is then initialised with the new parameters such as server IP address, game session ID, application server URL and tick rate.

Users can join the VR application by selecting the "Join as a Client" option in the launch menu, which requires a game session ID to be joined.

C. VR Application

After joining a session, users can interact in the VR application by sharing video content using the tablet component, available in the virtual room.

Users can hover their VR controllers on top of the Tablet Game Object to open a file browser where files are available for streaming. Presentation files (i.e. .ppt, .pptx) or video files (i.e, .mp4, .avi) can be played in the application. The chosen file is then uploaded to the application server with a multipart/form-data POST request. If the selected file is a presentation, the application server automatically creates a slideshow video from it, and responds back with the video URL. Once this process is completed, the PlayerNetworkLink class instructs the server to stream the video to all clients.

Algorithm 1 Simulated VR Users Mobility

```
NumberOfVRClients := e.g. 10
NumberOfMovement Loops := e.g. 10
NumberOfMovementSteps := e.g. 2
StepDuration := e.g. 150
Movement Directions := w,a,s,d keys
while (NumberOfVRClients client windows not spawned) do
    Spawn VR clientwindow and connect to test server
end
while (NumberOfMovementLoops > 0) do
    while (Not all client windows covered) do
        Activate client window
        Generate random movement direction
        Move in the direction for stepDuration
    end
end
```

IV. Results and Discussion

For testing purposes, multiple users scenarios were created with simulated users following various mobility patterns. This required the development of an automated VR user simulation mobility algorithm, presented in Algorithm 1. The simulation algorithm was developed using the AutoHotKey (AHK) tool, which simulates keyboard keys being pressed, moving the simulated users. The algorithm also needs the number of simulated clients, number of movement loops (the number of times a movement needs to be done in each client window), movement steps (the number of times the player object should move in each iteration) and movement duration (indicating for how long the movement key must be pressed), as configurable parameters. Fig. 3 presents the ability of the application to support multiple clients. A real user is viewing content from a VR headset in a computer (the content seen in the headset is mirrored on the bottom monitor of Fig. 3) and the other computer generates simulated clients, such as the one presented on the top screen.

Important QoS metrics were considered for testing. Throughput was analysed in three scenarios with 2, 5 and 10 VR users, with video content. In each scenario, one of these users is the server user, and the remaining ones are clients.

An extra scenario with 6 VR users was analysed in terms of tick rate (15Hz, 60Hz and 120Hz), and its impact on throughput and RTT.

The Windows 10 computer used in the tests has an Intel 8th generation i7 8200u processor with 16GB LPDDR3 of RAM. Wireshark was used for network analysis.

A. Throughput Analysis

For the throughput analysis, multimedia content is displayed in the virtual monitor in the application for all users. Mobility is also introduced for the simulated users, according to Algorithm 1, for a realistic throughput analysis.

The first scenario consists of two users (i.e. one server and a client). Fig. 4 contains a plot with the throughput of this scenario. The first peak (~28kbps) happens at the 15s mark

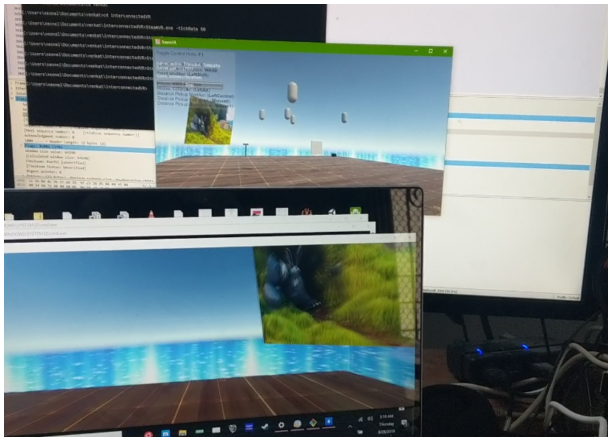

Fig. 3. Real and simulated users

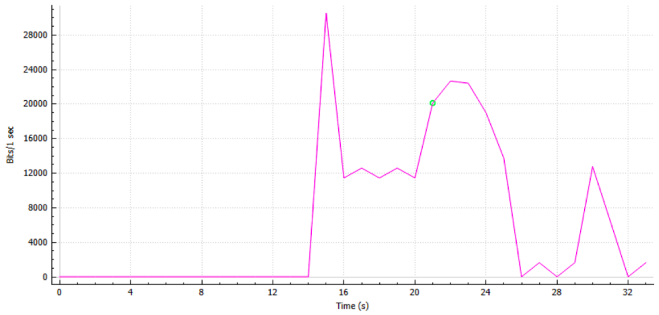

Fig. 4. Throughput for 2 users with video stream

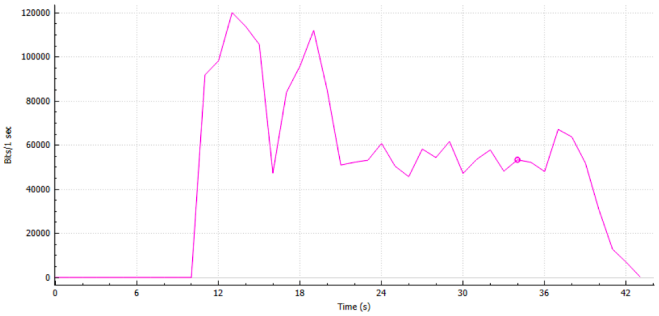

Fig. 5. Throughput for 5 users with video stream

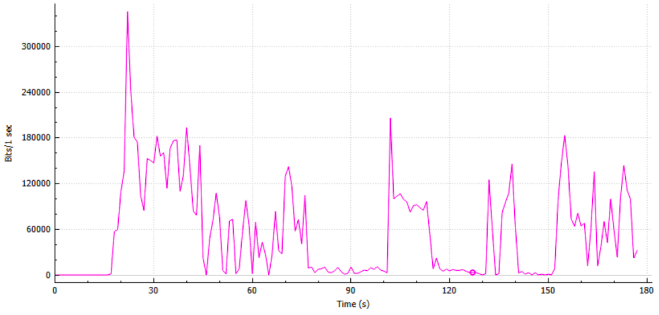

Fig. 6. Throughput for 10 users with video stream

in the graph, when the client connects. From 16s to 20s, (~12kbps) there was no user mobility. During the seconds 20s and 26s (~24kbps) the video stream and the mobility algorithm start, and a peak during seconds 29s-32s indicates

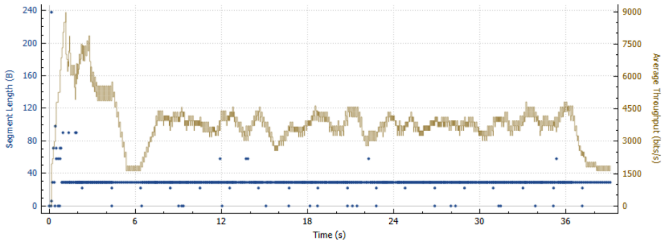

Fig. 7. Throughput for 15Hz Tick Rate

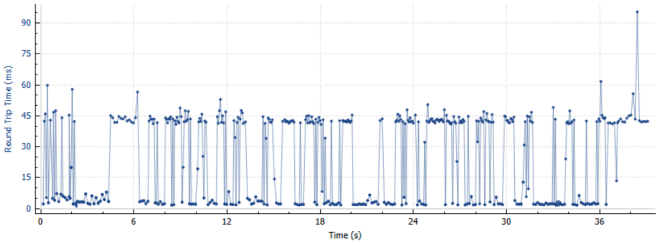

Fig. 8. RTT for 15Hz Tick Rate

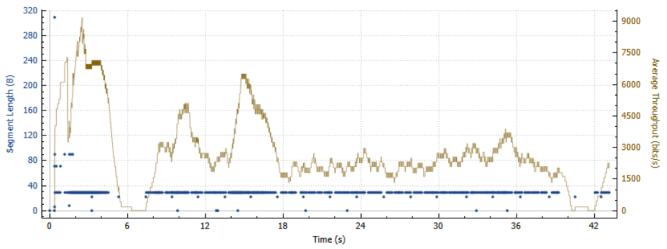

Fig. 9. Throughput for 60Hz Tick Rate

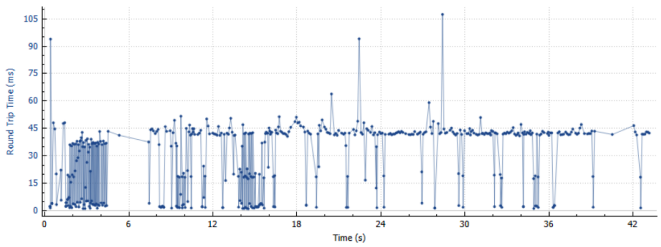

Fig. 10. RTT for 60Hz Tick Rate

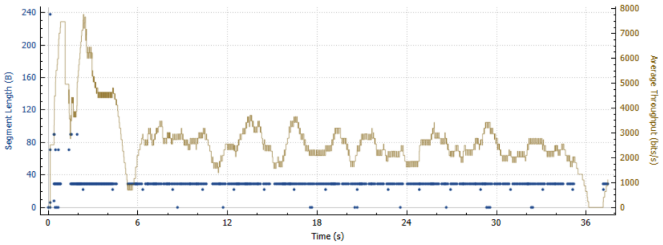

Fig. 11. Throughput for 120Hz Tick Rate

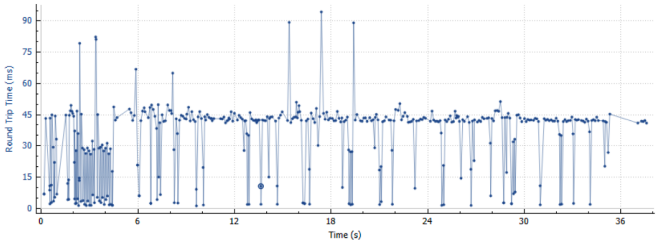

Fig. 12. RTT for 120Hz Tick Rate

video buffering.

The second scenario contains five users (i.e. one server and four clients). Fig. 5 illustrates the throughput of this scenario. The first peak (~120kbps) observed at the 13s mark indicates the clients being connected. From 16s to 20s (~110kbps), video buffering initiated. During the seconds 20s and 39s (~57kbps) the video stream and the mobility algorithm are running without any lags or decrease in quality.

The third scenario increases the number of users to ten (i.e. one server and nine clients). This scenario requires substantial processing power, as all users (i.e. virtual environments and mobility algorithm) are executed in the same computer, through simulation. Fig. 6 illustrates the throughput of this scenario. The first peak (~350kbps) observed at the 25s mark indicates the clients first connection. The mobility algorithm started at 45s and from 28s to 32s video content started playing. Three of the clients were able to play the video, while the others were buffering content. At the 77s mark, the platform became unstable with slowdowns until the 100s mark, and additional slowdowns at the 140s mark.

These experiments demonstrate that in scenarios with 10 or more simulated users, there is a decrease in quality, with long video buffering and slowdowns in the VR application. For this type of scenarios, a computer with higher specifications is required.

B. Tick Rate Analysis

The tick rate metric refers to the number of times the server synchronises the states with clients. For the analysis of the effect of different tick rates in throughput and RTT, which indicate network performance, three rates were selected: 15Hz, 60Hz and 120Hz. The scenarios contain six users: one server and five clients.

The first scenario consists of users with a tick rate of 15 times per second. The average throughput is ~4kbps, as illustrated in Fig. 7, with the minimum and maximum throughput ranging from ~1.5kbps to 9kbps. A peak of 9kbps is reached when the clients first connect to the server. From seconds 7s to 37s, the simulation algorithm is active. The RTT for this scenario (Fig. 8) varies from 5ms to 45ms for most of the execution time.

The second scenario increases the tick rate to 60 times per second. The average throughput is ~3kbps, as illustrated in Fig. 9, with the minimum and maximum throughput ranging from ~0.5kbps to 9kbps. This scenario also has a peak of 9kbps when the clients first connect to the server. From seconds 8s to 40s, the simulation algorithm is active. The RTT for this

scenario (Fig. 10) is more stable at 45ms with occasional 5ms bursts.

The third scenario increases the tick rate to 120 times per second, which is network-intensive. The average throughput is ~2.5kbps, as illustrated in Fig. 11, with the minimum and maximum throughput ranging from ~1kbps to 8kbps. This scenario has a peak of 8kbps is reached when the clients first connect to the server. From seconds 5s to 36s, the simulation algorithm is active. The RTT for this scenario (Fig. 12) is mostly at 45ms with occasional 5ms bursts.

These three scenarios demonstrate that the increase of the tick rate impact throughput and RTT. It is clear that the plots from Figs. 7 and 8 (15Hz) are more stable than the plots in Figs. 9-12 (60Hz and 120Hz), which represent a decrease in throughput stability and network performance.

V. Conclusions and Future Work

This paper presented the design, implementation and analysis of a multi-user VR application in a variety of scenarios. The application allows users to share multimedia content such as videos and presentations and interact with each other.

The application allows real and simulated VR users to be included in the scenarios, so the performance can be evaluated. An algorithm for VR user mobility was also introduced.

Tests indicated that in the computer used for the experiments, when the number of users is equal or greater than 10 simultaneous users, the performance is affected with longer buffering times and overall slowdowns. High tick rates, which are related to the smoothness of video, also affect throughput and RTT.

Overall, a multi-user application with real and simulated users is useful for performance testing, and it was possible to observe the scenarios that performed the best in a computer with the given specifications.

Future work includes the addition of other types of rich media content to be shared among users, and user prioritisation with different quality levels.

Acknowledgement

This work was supported by the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement no. 870610 for the TRACTION project. The support of the Science Foundation Ireland (SFI) Research Centres Programme Grant Numbers 12/RC/2289_P2 (Insight) and 16/SP/3804 (ENABLE) is gratefully acknowledged.

References

[1] A. A. Simiscuka, T. M. Markande, and G.-M. Muntean, "Real-Virtual World Device Synchronisation in a Cloud-enabled Social Virtual Reality IoT Network," *IEEE Access*, vol. 7, pp. 106 588–106 599, 2019.

[2] "Blender.org - Home of the Blender project - Free and Open 3D Creation Software," 2020. [Online]. Available: https://www.blender.org/

[3] T. I. Chowdhury, "Towards Reverse Disability Simulation in a Virtual Environment," in *25th IEEE Conference on Virtual Reality and 3D User Interfaces, VR 2018 - Proceedings*. Institute of Electrical and Electronics Engineers Inc., aug 2018, pp. 803–804.

[4] J. W. Park, F. S. Nahm, J. H. Kim, Y. T. Jeon, J. H. Ryu, and S. H. Han, "The Effect of Mirroring Display of Virtual Reality Tour of the Operating Theatre on Preoperative Anxiety: A Randomized Controlled Trial," *IEEE Journal of Biomedical and Health Informatics*, vol. 23, no. 6, pp. 2655–2660, nov 2019.

[5] C. H. Muntean, D. Bogusevschi, and G.-M. Muntean, *Innovative Technology-based Solutions for Primary, Secondary and Tertiary STEM Education*. Paragon Publishing, 2019.

[6] S. Abbas, A. A. Simiscuka, and G. M. Muntean, "A Platform Agnostic Solution for Inter-Communication between Virtual Reality Devices," in *IEEE 5th World Forum on Internet of Things, WF-IoT 2019 - Conference Proceedings*. Institute of Electrical and Electronics Engineers Inc., apr 2019, pp. 189–194.

[7] H. Lee, G. Ha, S. Lee, and S. Kim, "A mixed reality tele-presence platform to exchange emotion and sensory information based on MPEG-V standard," in *Proceedings - IEEE Virtual Reality*. IEEE Computer Society, apr 2017, pp. 349–350.

[8] S. Gunkel, M. Prins, H. Stokking, and O. Niamut, "WebVR meets WebRTC: Towards 360-degree social VR experiences," in *Proceedings - IEEE Virtual Reality*. IEEE Computer Society, apr 2017, pp. 457–458.

[9] F. Silva, D. Bogusevschi, and G. Muntean, "Innovative algorithms for prioritised ar/vr content delivery," in *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, 2018, pp. 1–5.

[10] D. Polancec and I. Mekterovic, "Developing MOBA games using the Unity game engine," in *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2017 - Proceedings*. Institute of Electrical and Electronics Engineers Inc., jul 2017, pp. 1510–1515.

[11] N. Singh and S. Singh, "Virtual reality: A brief survey," in *2017 International Conference on Information Communication and Embedded Systems, ICICES 2017*. Institute of Electrical and Electronics Engineers Inc., oct 2017.

[12] "Oculus Rift: VR Headset for VR Ready PCs | Oculus." [Online]. Available: https://www.oculus.com/rift/#oui-csl-rift-games=robo-recall

[13] "VIVE | Discover Virtual Reality Beyond Imagination." [Online]. Available: https://www.vive.com/eu/

[14] "PlayStation VR | The VR gaming system for PS4 | PlayStation." [Online]. Available: https://www.playstation.com/en-ie/explore/playstation-vr/

[15] J. Tham, A. H. Duin, L. Gee, N. Ernst, B. Abdelqader, and M. McGrath, "Understanding Virtual Reality: Presence, Embodiment, and Professional Practice," *IEEE Transactions on Professional Communication*, vol. 61, no. 2, pp. 178–195, jun 2018.

[16] "Unity - Manual: OpenVR." [Online]. Available: https://docs.unity3d.com/Manual/VRDevices-OpenVR.html

[17] "SteamVR." [Online]. Available: https://www.steamvr.com/

[18] "What is the Mixed Reality Toolkit | Mixed Reality Toolkit Documentation." [Online]. Available: https://microsoft.github.io/MixedRealityToolkit-Unity/README.html

[19] "VRTK - Virtual Reality Toolkit." [Online]. Available: https://vrtoolkit.readme.io/

[20] "Quickstart for Google VR SDK for Unity with Android." [Online]. Available: https://developers.google.com/vr/develop/unity/get-started-android

[21] "Unity Real-Time Development Platform | 3D, 2D VR & AR Visualizations." [Online]. Available: https://unity.com/

[22] N. Hossain, M. A. Hossain, M. Z. Hossain, M. H. I. Sohag, and S. Rahman, "OAuth-SSO: A Framework to Secure the OAuth-Based SSO Service for Packaged Web Applications," in *Proceedings - 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications and 12th IEEE International Conference on Big Data Science and Engineering, Trustcom/BigDataSE 2018*, sep 2018, pp. 1575–1578.

[23] "Electron | Build cross platform desktop apps with JavaScript, HTML, and CSS." [Online]. Available: https://electronjs.org/

[24] T. Kushida and Y. Shibata, "Framework of end-to-end performance measurement and analysis system for Internet applications," in *International Conference on Information Networking*, vol. 2001-January. IEEE Computer Society, 2001, pp. 674–679.

[25] A. A. Simiscuka and G.-M. Muntean, "Synchronisation between Real and Virtual-World Devices in a VR-IoT Environment," in *Proc. of the IEEE International Symposium on Broadband Multimedia Systems*, 2018, pp. 1–6.