A Platform Agnostic Solution for Inter-Communication between Virtual Reality Devices

Sami Abbas
School of Electronic Engineering
Dublin City University, Ireland
sami.abbas4@mail.dcu.ie

Anderson Augusto Simiscuka
School of Electronic Engineering
Dublin City University, Ireland
anderson.simiscuka2@mail.dcu.ie

Gabriel-Miro Muntean
School of Electronic Engineering
Dublin City University, Ireland
gabriel.muntean@dcu.ie

***Abstract*—Virtual Reality (VR) allows users to interact with intuitive environments to manipulate games, applications and even other Internet of Things (IoT) devices. VR devices have been brought to users by different vendors, supporting different development platforms and features. Many VR applications, however, do not support simultaneous VR devices from different vendors. The few applications that support this feature are not open to developers, with little information on their implementation and application performance. Using VRTK and Unity, this paper intends to present a VR application that supports multiple VR hardware platforms with the use of the VRTK SDK along with the appropriate VR hardware SDKs. This application supports multiple users in a VR environment where they can interact with each other and with objects in the VR space. Other features such as file transfer over the VR application are also supported. Such an application is important to allow developers to design future applications that support multiple devices regardless of the vendor, creating convenient multiuser VR applications with less limitations.**

***Index Terms*—VR, inter-communication, application, interface**

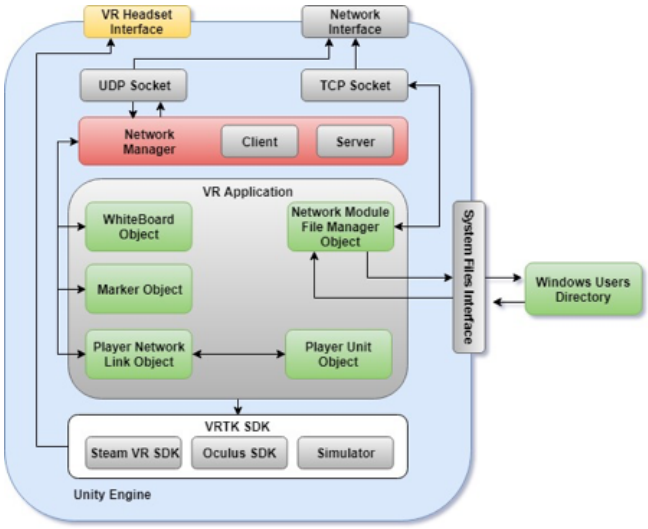

Fig. 1. VR Block Diagram

I. Introduction

Virtual Reality (VR) platforms are becoming a mainstream technology and affordable to many users, resulting in many VR applications being developed [1]. However, most of these applications do not allow users of different VR hardware platforms to interact with each other in the same VR application. This is because the developers tend to design their application to a specific platform that holds most of the market share such as HTC Vive and Oculus Rift [2]. Some VR applications are platform exclusive which is normally established through a mutually-beneficial contract between the VR platform company and the developer of the client application. This limits the end user access to applications that are only developed to specific vendors.

This paper proposes the design and implementation of a VR application that allows multiple users to use different VR hardware platforms on separate computers to interact with each other in a universal virtual environment over a network. This means that this VR application supports multiple users on different hardware platforms. This integrated environment is aligned with the concept of seamlessly interconnecting devices in an Internet of Things (IoT) network that also includes VR. Developers can benefit from this approach to incorporate the support of all platforms in future applications.

An approach to address the differences among VR platforms is to develop a single software development kit (SDK) that can communicate between the VR platform SDKs and the VR application itself. This means that developers only have to write the application code once and create an interface with the SDK toolkit. Then the toolkit can interface with any number of the VR SDKs that the developer wishes to support. A popular toolkit for this purpose is called Virtual Reality Toolkit (VRTK) [3]. VRTK, however, does not support networking. Unity, one of the major game development engines in the industry, has the capability of supporting networking in VR applications [4]. Its support for high quality 2D and 3D video games make it one of the leading development engines in VR technology.

The proposed VR application supports multiple VR hardware platforms such as the HTC Vive and Oculus Rift. A

978-1-5386-4980-0/19/$31.00

shared VR environment is included, where users can interact with each other. Objects such as a shared whiteboard and a marker tool are also implemented in aplication. The application is also able to handle the sending/receiving of files between each machine connected to the host. This application must have a type of network communication link between each machine to support the transmission of data.

Using the VRTK toolkit and Unity engine, the application can be built once and then linked to the official SDKs provided by Oculus and HTC. Unity is be the Integrated Development Environment (IDE) used for the development of the VR application.

Network tools provided by the Unity platform are used to establish a communications pipeline between the VR applications. Socket programming is used to transfer the files across the network between the two VR application instances. The diagram presented in fig. 1 shows the block layout of the modules that are included in this VR application.

This paper is organised as follows. In section II related works are presented and section III introduces the technical description of the application. Section IV details the performance analysis, and section V presents conclusions and future work directions.

II. Related Works

There is a wide range of IoT services, from low bandwidth smart metering to high bitrate rich media applications. At the same time there is a large number of rich media-enabled IoT devices (i.e. there are more than 0.2 million AR/VR devices globally), which generate a massive amount of data traffic at data rates, often at more than 1 Gbps and with an expected latency of less than 1ms [5]. These services use diverse solutions to enable data transmission at high quality, including network selection [6], load balancing [7], personalised content adjustment [8] and adaptive delivery [9]. More recently there has been a significant increase in demand for very high bitrate rich media services, including VR, in general context and in particular that of IoT [10].

VR and IoT have also been integrated in order to maximize the high inter-operability of IoT services with the intuitive nature of VR [11], [12]. This integration also allowed the creation of an immersive virtual representation of a smart city with remote sensing [13]. These VR-IoT integrated platforms can provide a customizable and user-friendly environment for controlling real IoT devices in virtual environments.

An approach for exchanging avatars for collaborative VR systems was presented in [14]. The focus relies on the transfer of emotions in real-time by using a software- independent data representation.

Authors in [15] explore hardware interactivity and VR devices through two games designed to use the Oculus Rift SDK technology with alternative methods of hardware for communication.

III. Technical Description

A. VR Room

The VR room accommodates the users in the VR space. It allows users to interact with each other or with other objects. It consists of a ground plane and a set of four walls that encloses the room to prevent the users from going too far and falling off the edge of the 3D environment. The room was created using the Unity 3D editor to create 3D shapes and move the objects around the room. Fig. 2 shows the overall room layout.

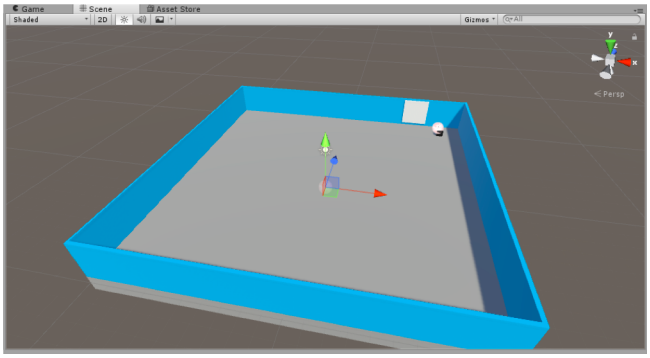

Fig. 2. VR Room

B. WhiteBoard Objects

A set of objects consisting of a whiteboard, table and marker were created for use in the VR room. Users can pick up the marker from the table and write on the whiteboard to communicate with each other in this VR space. The other user wearing a different VR headset is able to see the changes made on the board. Fig. 3 shows the whiteboard object in the VR room. Oculus Rift and HTC Vive provide controllers for users to manipulate the application.

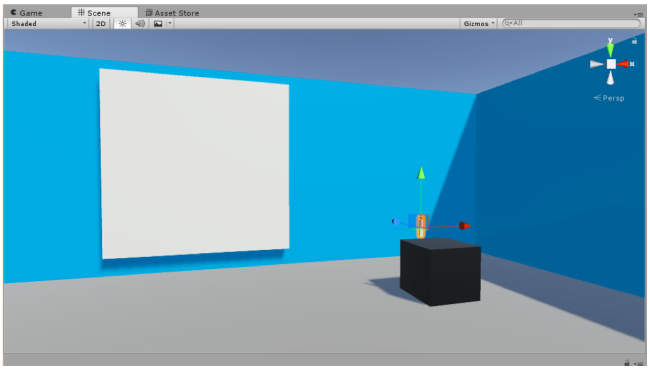

Fig. 3. Whiteboard Objects

The whiteboard and the marker have C scripts attached to them to enable the intended functions of the board. Once the user picks up the marker the scripts start.

First, the marker script finds the current position of the marker in the VR room. Then it gets the distance between the marker tip and another object in the VR room. As soon as the marker collides with another object it is able to record

Algorithm 1 Marker Behaviour

Require: $marker\_position$; $marker$; $whiteboard\_object$
Output : $whiteboard\_x\_y\_position$; $whiteboard\_touch$; $marker\_rotation\_lock$; $last\_hit$; $server\_touch$
if *(marker hits an_object)* **then**
  if *(an_object == whiteboard_object)* **then**
    set $whiteboard\_x\_y\_position$
    $whiteboard\_touch$ = true
    $marker\_rotation\_lock$ = true
    return $whiteboard\_x\_y\_position$, $whiteboard\_touch$, $marker\_rotation\_lock$
  end
end
else
  $whiteboard\_touch$ = false
  $last\_hit$ = false
  $server\_touch$ = false
  if *(marker != touch)* **then**
    $marker\_rotation\_lock$ = false
  end
end
return $whiteboard\_touch$, $last\_hit$, $server\_touch$, $marker\_rotation\_lock$

Algorithm 2 Whiteboard Behaviour per Frame

Require: $server$; $marker\_touch$
Output : $marker$; $whiteboard$
if *(server changes)* **then**
  $server$.cmd $client$ with new $server.x, y\_position$
  if *(skip_over == true)* **then**
    draw x, y_points from $server$
  end
  save $server.x, y\_position$
  apply new textures to whiteboard
  reset change and StepIn if statements
end
get $marker.position$
if *(marker == touch)* **then**
  draw $x, y\_position$
  notify $server$ of updates
  apply new textures to whiteboard
end
save $marker.x, y\_position$
save $marker.touch$
return $marker$, $whiteboard$

the exact position of the marker. It, then, checks if the object it has hit was the whiteboard. If it is, then it informs the whiteboard at what point (x, y) the marker has been pressed. The whiteboard, then, starts recording the information written on the board. It also locks the rotation of the marker so that when the user moves their hands the marker stays perpendicular to the whiteboard. This prevents the marker from passing through the whiteboard object.

When the marker stops touching the whiteboard, it informs the whiteboard it is no longer touching it. It also communicates to the server that the marker is no longer touching the whiteboard allowing other users to draw. On the whiteboard script, the colour of the marker is first be set to the color black. The reason it is controlled on the whiteboard and not in the marker is to allow marker to simply find the x, y coordinates when it collides. The rest is be handled by the whiteboard script.

Updates on the whiteboard are sent by the server to all clients. If there is an update, the x and y coordinates points are sent by the server, painting the pixels on the 2D texture plane of the whiteboard. When the marker is touching the board, the pixels of that point can be painted on the 2D texture plane. The server updates all clients with changes on the whiteboard.

Algorithms 1 and 2 represent the behaviour of the marker and whiteboard, respectively.

C. Player Avatar Unit

Avatars represent the players in the VR room. They enable each local player to see the location of the remote player in the VR room. Avatars are solid objects of a capsule style structure that represent the players position in the room. This object follows the location of the player wherever they are in the room, as seen in fig. 4.

Fig. 4. Player Avatar Unit

D. Network Manager

The network manager has several responsibilities inside the VR application. It has a UI so that the user can select whether it wants to host as a server or connect as a client to another VR instance. It also has the responsibility to spawn the player units in the room and to track their positions across all clients and to issue new position updates when needed.

It also handles whiteboard updates, this is done when the user communicates commands to the server with the new whiteboard (x, y) position points updates that need to be marked on all clients. The server, then, relays this information to all clients, as seen in the diagram of fig. 5.

The Network Manager uses Unity's Server command and Client RPC protocol. When the user issues an update it communicates with the server to do the task. The server then executes the update. It also synchronises all clients using the RPC functions with the updated parameters, using the UDP protocol. If the player position is lost (UDP packet dropped), it is unimportant, as each time the player moves slightly, new updates are sent, so the VR application can refresh its current position with the new information. Regarding the whiteboard, a function joins all points on the board as one line to make

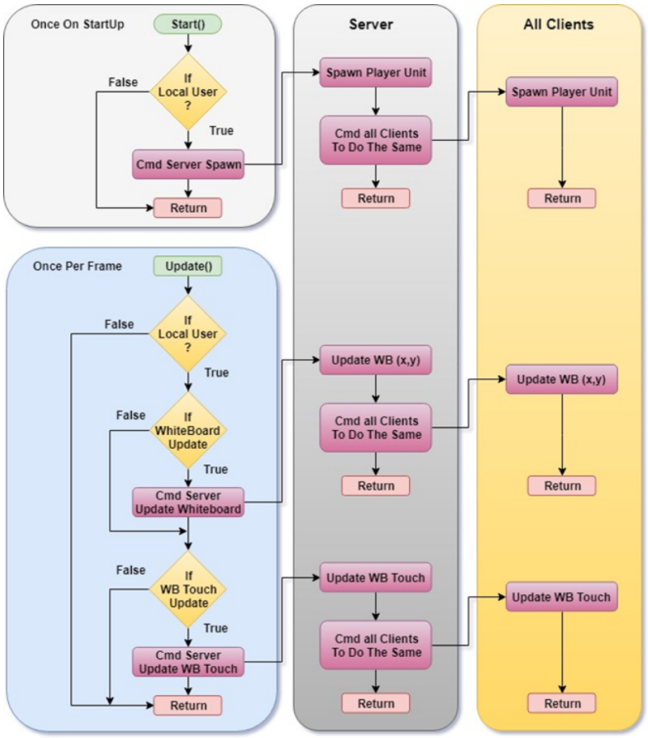

Fig. 5. Network Manager Flow Chart

the overall experience smooth which results in users drawing continuous lines.

E. File Manager

The file manager script from fig. 6 is responsible to send and receive files between each VR instance. The script itself setup the directories in Windows on the VR application startup. This means that users can simply drag files into the appropriate output folder for possible data transmissions.

A script is activated when the user pushes the M button for sending or N button for receiving files. When the M button is pressed, the user intends to send files. The file selected is sent to a buffer and the server waits for a client to connect to receive the files. When the N button is pressed, it connects to a new waiting server. The data is then transmitted over the network. After file transmission, both programs at each side end connection and close the sockets. Threads are used for this to prevent the VR application from freezing up.

The maximum permitted size that can be sent over is 2.1 GB due to a signed 32-bit integer value holding the number of the file size in bytes.

IV. Performance Analysis

Figs. 7, 8 and 9 demonstrate users testing the controls and the whiteboard and the screen view of the application. Other tests related to the network performance when running the platform were executed. Wired and wireless sessions were analysed, as well as the file transfer feature.

Results for the wired session are available in fig. 10. It can be seen that the peaks of the graph (from 50s to 125s)

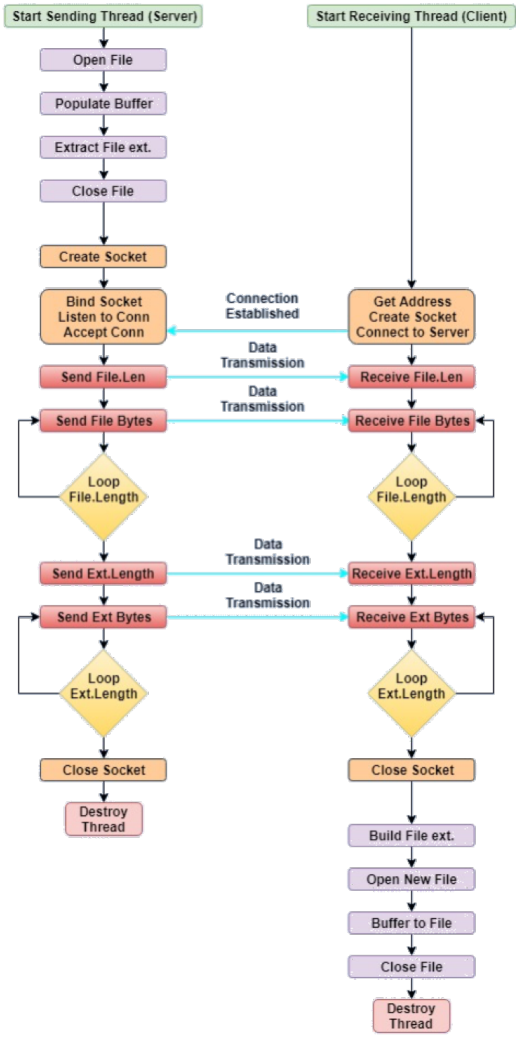

Fig. 6. File Manager Flow Chart

Fig. 7. Controls Testing

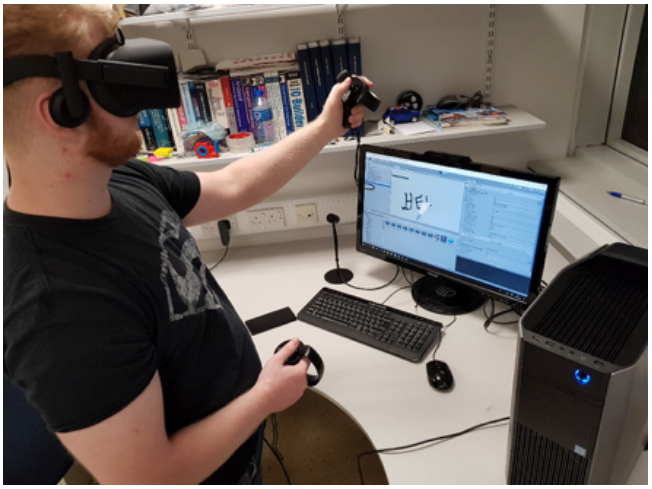

Fig. 8. Whiteboard Testing

Fig. 9. Screen View of the Whiteboard

show the users moving around the room very quickly, requiring more data to be transferred. It reduces when they approach the whiteboard and start to draw on the whiteboard. This test resulted in less than 1% utilization of the 1Gbps link. The maximum bandwidth consumed on the link was 0.12 Megabits per second.

The wireless session results are shown in fig. 11. The overall speed is reduced due to the payloads of the wireless TCP packets being larger than the wired TCP packets. The speed reduction can also be caused by the communications link that requires a 3-way TCP handshake. Maximum speed in wireless link was 0.104 Megabits per second.

Wired file transfer results are shown in fig. 12. The first bump in the graph is related to a file being sent the second bump related to the file being received. The maximum speed achieved was 2.632 Megabits per second.

Wireless file transfer results are available in fig. 13. The maximum speed was 1.456 Megabits per second.

Table I shows both the wired and wireless file transfer results. Sending files over the wired and wireless network took

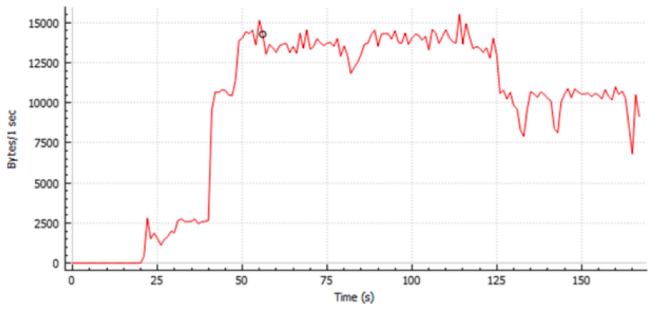

Fig. 10. Wired Test Results

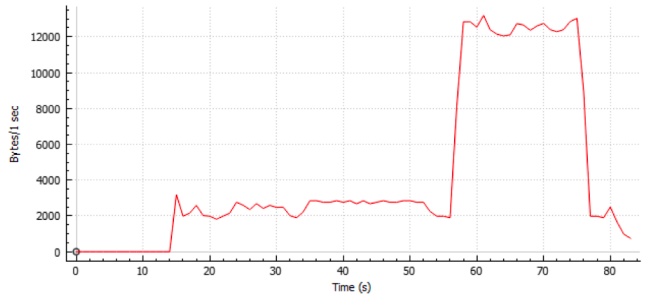

Fig. 11. Wireless Test Results

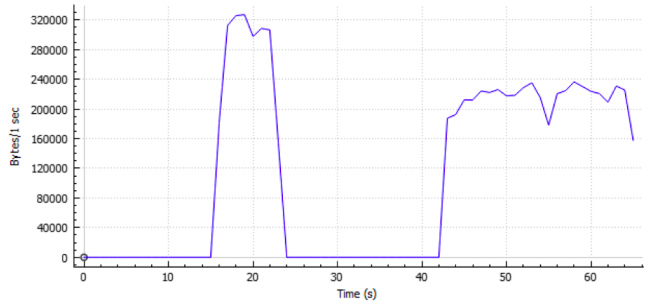

Fig. 12. Wired File Transfer Test Results

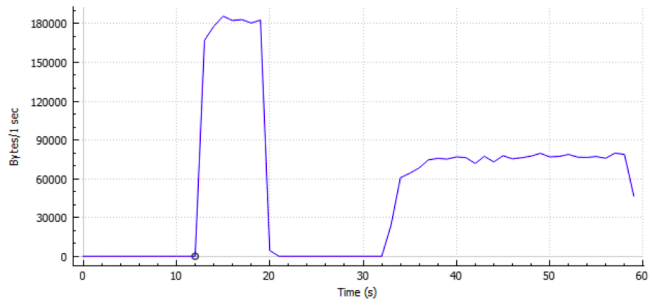

Fig. 13. Wireless File Transfer Test Results

approximately the same amount of time. The overall speed in the wireless network, however, was lower than in the wired network.

The VR application was also tested in a 3 user scenario. The network performance results can be seen in fig. 14. The network usage was less than 1% of the links resources.

TABLE I
PEAK SPEEDS

	Wired Connection		Wireless Connection	
	Time (S)	Peak Speed (Bytes/S)	Time (S)	Peak Speed (Bytes/S)
Asus Send to Dell	9	329000	9	182000
Dell Send to Asus	23	235000	27	80000

Between seconds 36s and 48s, all three users were using the whiteboard one after another. Fig. 15 shows the 3 users avatars in the shared VR room.

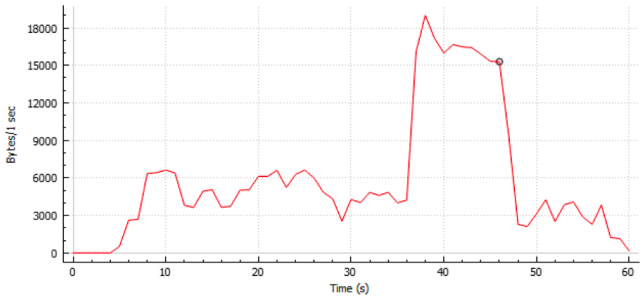

Fig. 14. Network Test 3-Player Session

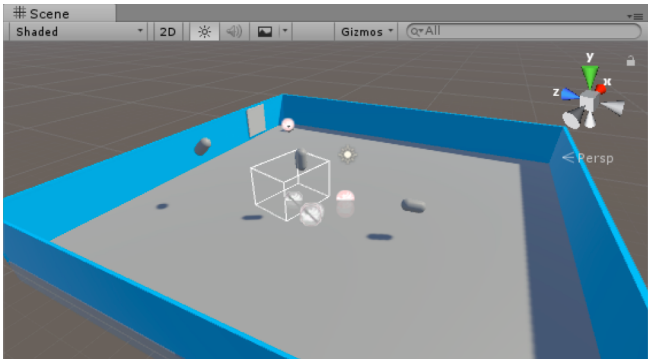

Fig. 15. Avatar Units for 3 Players

V. CONCLUSIONS AND FUTURE WORK

This paper presented the design and implementation of a VR application that allows multiple users to use different VR hardware platforms on separate computers to interact with each other in a universal virtual environment over a network. Oculus Rift and HTC Vive headsets were employed in the testbed which includes an application that work seamlessly in both platforms with simultaneous users.

The overall VR application load on the network was significantly low as results demonstrated, which means that the network could support many instances of the VR application provided that the computer can handle the graphical load of the VR application.

Future work includes improvements on the file transfer module, which can use multi-threaded sockets to speed up the transfer process. Other textures can be added to the VR room to make it more appealing. More efforts can also put into support for additional VR hardware platforms.

ACKNOWLEDGEMENT

This work was supported by the Irish Research Council and Dublin City University, grant number EPSPG/2015/178, and in part by Dublin City University Entwine Research Centre and European Union's Horizon 2020 Research and Innovation programme under Grant Agreement no. 688503 for NEWTON project (http://newtonproject.eu).

REFERENCES

[1] N. Singh and S. Singh, "Virtual Reality: A Brief Survey," in *Proc. of the International Conference on Information Communication and Embedded Systems (ICICES)*, Feb. 2017, pp. 1–6.

[2] M. Suznjevic, M. Mandurov, and M. Matijasevic, "Performance and QoE assessment of HTC Vive and Oculus Rift for pick-and-place tasks in VR," *2017 9th International Conference on Quality of Multimedia Experience, QoMEX 2017*, pp. 1–3, 2017.

[3] G. Regal, R. Schatz, J. Schrammel, and S. Suette, "VRate: A Unity3D Asset for integrating Subjective Assessment Questionnaires in Virtual Environments," *2018 10th International Conference on Quality of Multimedia Experience, QoMEX 2018*, pp. 1–3, 2018.

[4] M. Hubbell and J. Kepner, "Large scale network situational awareness via 3D gaming technology," *2012 IEEE Conference on High Performance Extreme Computing (HPEC)*, pp. 1–5, 2012.

[5] P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila, and T. Taleb, "Survey on multi-access edge computing for internet of things realization," *IEEE Communications Surveys Tutorials*, pp. 1–1, 2018.

[6] O. Ormond, G.-M. Muntean, and J. Murphy, "Network Selection Strategy in Heterogeneous Wireless Networks," *Information Technology and Telecommunications Conference (ITT)*, October 2005.

[7] A. Hava, Y. Ghamri-Doudane, G.-M. Muntean, and J. Murphy, "Increasing user perceived quality by selective load balancing of video traffic in wireless networks," *IEEE Transactions on Broadcasting*, vol. 61, no. 2, pp. 238–250, June 2015.

[8] C. H. Muntean and J. McManis, "A qos-aware adaptive web-based system," in *IEEE International Conference on Communications*, vol. 4, June 2004, pp. 2204–2208.

[9] G.-M. Muntean, "Efficient delivery of multimedia streams over broadband networks using qoas," *IEEE Transactions on Broadcasting*, vol. 52, no. 2, pp. 230–235, June 2006.

[10] "Cisco Visual Networking Index: Forecast and Trend," 2017-2022. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.pdf

[11] M. I. Choi, L. W. Park, S. Lee, J. Y. Hwang, and S. Park, "Design and implementation of Hyper-connected IoT-VR Platform for customizable and intuitive remote services," *2017 IEEE International Conference on Consumer Electronics, ICCE 2017*, pp. 396–397, 2017.

[12] 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.

[13] Z. Lv, T. Yin, H. Song, and G. Chen, "Virtual Reality Smart City Based on WebVRGIS," *IEEE Internet of Things Journal*, vol. 4662, no. c, pp. 1–1, 2016.

[14] R. Klauck, S. Lorenz, and C. Hentschel, "Collaborative work in VR Systems: A software-independent exchange of avatar data," *IEEE International Conference on Consumer Electronics - Berlin, ICCE-Berlin*, vol. 2016-October, pp. 133–136, 2016.

[15] M. Mentzelopoulos, F. Tarpini, A. Emanuele, and A. Protopsaltis, "Hardware interfaces for VR applications: Evaluation on prototypes," *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing Hardware*, pp. 1578–1583, 2015.