Prototype of a Hybrid Communication System: Bridging VR and Video Conferencing

Marek Ďurica, Marek Vančo
Faculty of Electrical Engineering and Information Technology, Slovak University of Technology, Bratislava, Slovakia
marekdurica229@gmail.com, marek_vanco@stuba.sk

***Abstract*— The Metaverse and immersive technologies related to it open up new possibilities for remote communication and collaboration, but often create a barrier between users in virtual reality (VR) and those using traditional video conferencing tools. This work presents the design and implementation of a prototype hybrid communication system that addresses this problem by creating a bidirectional visual bridge. The system, developed in the Unity engine, allows users in VR to see participants of a standard video call on virtual panels, and conversely, video call participants see the view from the VR scene. It utilizes a combination of the Mirror library for synchronizing the state of the multiplayer VR scene and the Agora RTC SDK for real-time audio and video transmission between VR and non-VR clients, including sending custom video and audio sources from Unity. The result is a prototype demonstrating the technical feasibility of the proposed approach towards more flexible hybrid communication.**

Keywords—Metaverse, virtual reality, video conferencing, Unity Engine, Agora RTC

I. Introduction

In recent years, there has been growing interest in the concept of the Metaverse – a vision of persistent, shared virtual worlds that extend the capabilities of the internet, which was first described in the science fiction novel „Snow Crash“ by writer Neal Stephenson [1]. Virtual reality (VR) plays a key role in this vision, offering a high degree of immersion and sense of presence. Social and collaborative VR platforms like VRChat [2] or Meta Horizon Workrooms [3] demonstrate the potential of VR for remote interaction.

Despite this, widespread adoption of VR is still limited by hardware requirements, cost, and a general lack of appealing content for users [4]. Most professional and personal remote communication takes place via traditional video conferencing platforms (e.g., Zoom [5], Microsoft Teams [6], Google Meet [7]). While these platforms are accessible, they lack the immersiveness and quality of social interaction found in VR [8]. This creates an open window for hybrid solutions that would enable effective communication between users with different equipment – some in VR, others on standard devices. Recognizing this gap, academic research has been actively developing innovative, non-commercial systems that push the boundaries of hybrid collaboration. For instance, some solutions like **MRMAC** demonstrate how remote VR users can be virtually present and collaborate in a real-world space captured by 360° cameras, interacting with local users in a mixed reality environment [9].

This project focuses on this problem and presents the design, implementation, and evaluation of a prototype system that creates a bidirectional "window" between a VR environment created in the Unity engine and a standard video call. The contribution of this work is the demonstration of the technical feasibility of such a connection using a combination of the Mirror networking library [10] for VR scene synchronization and the Agora RTC platform for cross-platform A/V transmission. The article will discuss related and similar works, the architecture and design of such a system, key aspects of the implementation, and finally, presentation of the results.

II. Related Work

Existing communication solutions can be divided into two main categories. On one hand, there are native VR platforms (e.g., VRChat [2], Rec Room [12], Meta Horizon Worlds/Workrooms [3]), which provide immersive experiences for VR users but often function as closed ecosystems with limited or even no possibility of interaction for non-VR users.

On the other hand, research and development focuses on hybrid approaches. These include asymmetric visualization (displaying video of non-VR users in VR and vice versa), avatar representation of non-VR users in VR, shared interactive tools, and platform integrations, like Microsoft Mesh, introduced in the year 2023. However, creating a seamless and intuitive connection still faces challenges in the areas of interaction asymmetry, perception, and technical compatibility. The work **WebTransceiVR** [11] represents a relevant example of recent research in this area. Their proposal is an asymmetric collaboration toolkit for Unity that allows multiple non-VR users to connect to the shared virtual space of a VR user via a web browser. Their system distinguishes between interactive co-hosts and passive spectators, using WebRTC and HLS streaming for data transmission.

This project falls into the category of asymmetric visualization, providing a specific technical solution for bidirectional video/audio data flow using Agora and Unity.

III. Architecture and Design

The proposed system consists of two main client types and two communication layers, as well as a dedicated token server to connect the two client types. (see Fig. 1).

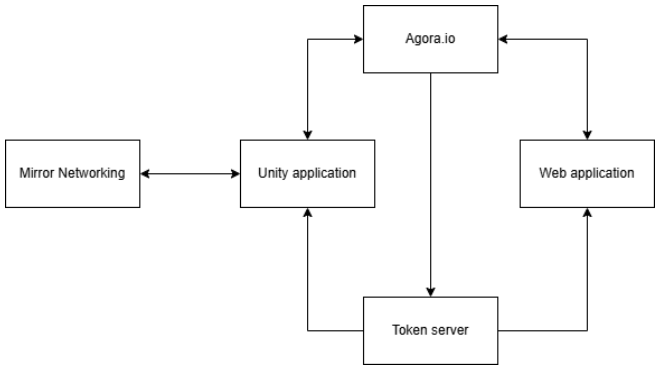

Figure 1. Simplified architecture of the hybrid system.

A. Unity application (VR client)

An application running on a PC with a connected VR headset (e.g., Meta Quest 2, which was used for testing). Unity was chosen for its long-standing, robust support for various VR platforms, most notably through the XR Interaction toolkit used in this project, as well as its flexibility in integrating third-party SDKs. This part of the architecture is responsible for:

- Rendering the VR scene and processing input from VR controllers (using Unity [13], OpenXR, XR Interaction Toolkit [14]).
- State synchronization (avatar position, interactions) with other VR clients in the same room using the **Mirror** library.
- Connecting to the **Agora RTC** [14] channel.
- Capturing the view from a defined camera (*sceneCamera*) in the Unity scene into a *RenderTexture*.
- Sending the *RenderTexture* content as an *ExternalVideoFrame* via the Agora SDK.
- Receiving video streams from non-VR clients via the Agora SDK.
- Displaying received video streams on UI panels (*RawImage* with *VideoSurface*) in the VR space.

B. Web application (Non-VR client)

A web application running on a server that users can connect to from a computer or a phone. Responsible for:

- Connecting to the same **Agora RTC** channel as the VR app using a compatible token.
- Sending a video stream from the user's webcam and an audio stream from the microphone via the Agora SDK.
- Receiving and displaying the video stream from the VR scene (sent by the VR client) and streams from other non-VR clients.

C. Token server

To enable connection between VR and non-VR clients, compatible Agora RTC tokens are required. These tokens are generated on demand by a server-side PHP script, which is hosted on a publicly accessible external server. This script utilizes Agora's official RtcTokenBuilder2 library. It issues tokens with a User ID (UID) of 0, which Agora interprets as allowing any user, and sets a token expiration time of 3600 seconds (1 hour). After this period, a new token must be generated for continued communication. The server responds to token requests with a JSON object containing the unique token, which is then acquired by both the Unity application and the web application.

D. Communication layers

- **Mirror Networking:** The Mirror library provides multiplayer for an unlimited number of users in a single virtual room. It synchronizes states between VR clients. The library was chosen over the Photon library [15] because it has no limit on the number of concurrent users.
- **Agora RTC:** Ensures A/V communication between the VR client(s) and non-VR clients via the global Agora SD-RTN™ network [16]. While there are many other similar libraries, Agora was chosen for its robust documentation, ease of use and vast compatibility.

IV. Implementation

The prototype implementation was carried out in Unity version 6.000.24f1, in the C# programming language. The Mirror, UnityXR, and Agora SDK libraries were used. Key points:

A. Mirror Setup

The standard *NetworkManager* was used to setup the Mirror multiplayer library for connection management and avatar synchronization using the *NetworkIdentity* and *NetworkTransform* components. Users in the virtual room can thus see each other. The *NetworkManager* object also serves as a 'spawn point' that creates newly connected users. The user interface visible when first joining the application is based on the default *Network Manager HUD*, but with buttons which are interactable by VR users. This is achieved by creating a Canvas object, setting it to display in "World Space" and positioning the user's initial spawn point in front of it. The user can later leave the network by interacting with the door placed in the virtual room, which activates the *NetworkManager.singleton.StopClient()* function.

B. Agora RTC Integration

The *JoinChannelVideo.cs* code serves as the main framework for the Agora RTC implementation. It can be split into these steps:

- **Authentication***:* Obtaining temporary tokens via an HTTP request to an external PHP script located on a server. This ensures that both the web and Unity applications receive a mutually valid token and can interlink.
- **Initialization***:* Setting up *RtcEngineContext* and its attributes (*AppID*, channel profile) and configuring the video encoder (*SetVideoEncoderConfiguration*).
- **Sending the VR View***:* Achieved by using the function *SetExternalVideoSource*. In the *Update()* method, the content of the RenderTexture (*sceneCamera*) is converted to *byte[]* (an array of bytes) and sent as an *ExternalVideoFrame* using *PushVideoFrame* to a custom video track which we

have created using *CreateCustomVideoTrack*. The view being sent originates from the *sceneCamera* object located within the scene. VR users can freely manipulate with it (XR Interaction Toolkit), which allows real-time movement and capturing of the virtual image.

- **Receiving the Video Call**: Processing *OnUserJoined* in *IRtcEngineEventHandler*. Dynamic creation of a UI GameObject (*RawImage*, *VideoSurface*). *VideoSurface* renders the received stream onto given monitors.

This code is placed within a manager empty game object as a component. The *SecondMonitor.cs* code extends the functionality of the *JoinChannelVideo.cs* code. It contains the field *public uint targetUID*, which is specified for each monitor in the scene, allowing each monitor to display a different user's video stream and dynamically update it.

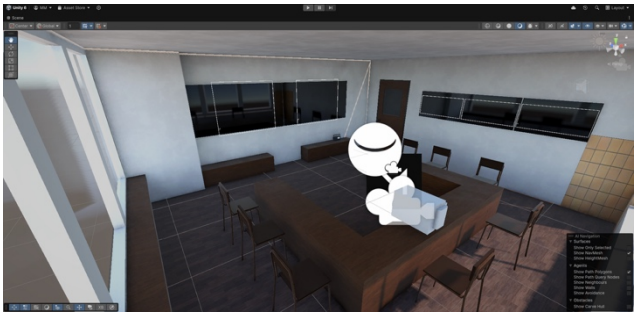

Figure 2. Appearance of the Unity room, based on room B405, FEI STU.

C. VR Interactions

The XR Interaction Toolkit library [14] was used for VR interactions. The user can move in the VR space using either teleportation or the joystick of his left-hand controller and interact with objects in the room. The XR Origin prefab (the player's character) has the *XR Origin*, *Input Action Manager* and *XR Input Modality Manager* components with motion controllers set up to track hand movement. A simple player avatar created in the MakeHuman software is set as the rig. The *PlayerIK.cs* code controls the movement of the avatar's arms and head through Inverse Kinematics.

D. Non-VR Client

The non-VR client was implemented as a simple HTML/CSS web page built on the Agora Web SDK. The user interface, defined in index.html, allows users to join an Agora session. The functionality in index.js uses the Agora RTC SDK to create a client that can join a channel with one of the predefined user IDs corresponding to the monitors in the Unity room. After connecting, the client initializes and publishes the local audio and video stream from the user's webcam and microphone. At the same time, it subscribes to and displays the video streams of other participants in the call, including the Unity scene, which is assigned a special UID. The token is downloaded automatically from the same source as for the Unity application. For the prototype, there are five connection buttons available and pressing any of them disables the others until the user disconnects. Attempting to connect to a monitor that is already occupied will not let the client through.

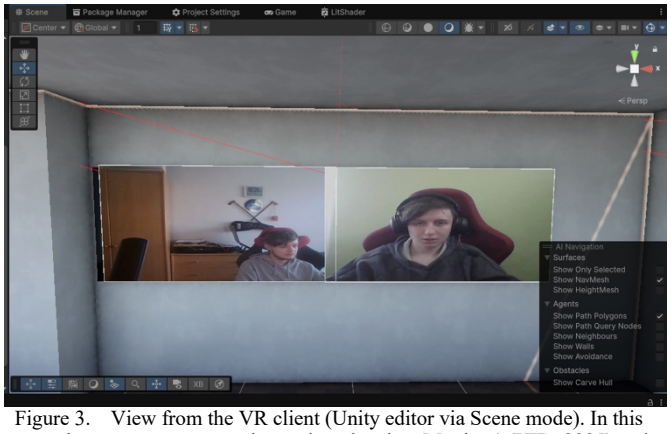

Figure 3. View from the VR client (Unity editor via Scene mode). In this example, users are connected to and rendered on Monitor1 (UID: 3395) and Monitor2 (UID: 3396).

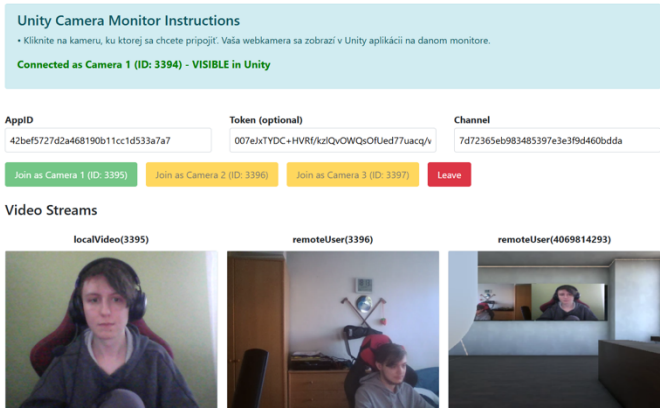

Figure 4. View from the web application.

The virtual room in Unity was modeled in Blender as a simplified representation of room B405 from the FEI STU building premises. A camera object *sceneCamera* is placed in the scene (see Fig. 5). This object is not static; the VR user acting as the session host can freely manipulate it and place it anywhere within the virtual environment. The view from this camera is captured and streamed. In addition to this main streaming camera, there are five monitors in the room - Monitor1 to Monitor5 - on which the video streams of VR users are rendered via the SecondMonitor.cs code. To enrich the social atmosphere, a 'Jukebox' object was also added to the scene, allowing music audible to all participants in the call to be played (see Fig. 6).

V. Results

The result of the work is a functional prototype demonstrating the proposed concept of hybrid communication. The prototype allows the user in VR to see a live video stream from the non-VR user on a UI panel as well as conversely display the VR scene to non-VR users by capturing a live video stream. The system uses Agora RTC for A/V transmission between the Unity and web clients and basic multiplayer was implemented with the Mirror Networking library and its NetworkManager functions.

With testing, we were able to confirm the functionality of video transmission in both directions over a standard internet connection without any severe performance issues. The application supports a maximum amount of five real-life streams at once and future work could focus on improving this scalability, as well as testing whether the application remains

sustainable with a higher number of streams. Video quality could also be improved, as the displays themselves are flat panels and are prone to distortion when viewed under certain oblique angles. On the side of the non-VR client, a better way of indicating that a certain monitor is already occupied could help make the UI more readable, as the current setup does not provide any immediate feedback to the fact.

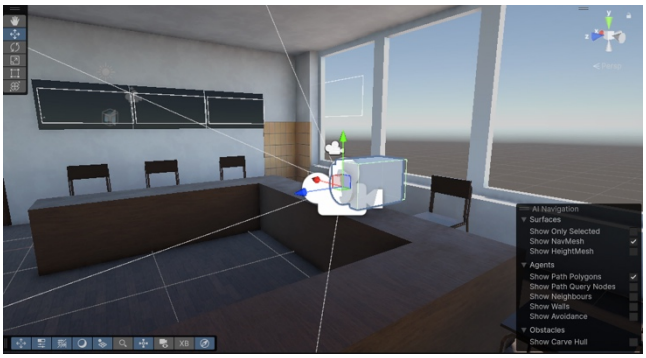

Figure 5. The *sceneCamera* object.

VI. Conclusion

This work addressed the challenge of connecting virtual environments with traditional video conferencing tools. The prototype presents the design of a hybrid communication system aiming to create a seamless bidirectional visual and audio bridge between users in virtual reality (VR) and participants connected via a standard web application.

The work demonstrates the technical feasibility of the proposed architecture based on the combination of the Unity engine, the Mirror networking library [10], and the Agora RTC SDK platform [15]. The implemented prototype successfully allows non-VR users to see a dynamic view from the VR scene captured by a movable camera controlled by the VR user and simultaneously allows VR users to see live video streams of non-VR participants on virtual monitors within the scene.

A. Potential Improvements

Future work could mainly focus on expanding interactivity between VR and non-VR users beyond video and audio display by adding various activities into the scene, for example by implementing a shared whiteboard where all users can write. Other directions include support for multiple independent rooms and custom avatars that users could import and use.

Acknowledgement

This paper was supported by DISIC (09I05-03-V2), NEXT (ERASMUS-EDU-2023-CBHE-STRAND-2), EULiST (ERASMUS), InteRViR (VEGA 1/0605/23).

References

[1] N. Stephenson, *Snow Crash*. Bantam Books, 1992.

[2] "VRChat," 2024. [Online]. Available: https://hello.vrchat.com/. [Accessed: 2025-04-27].

[3] "Horizon Workrooms Virtual Office and Meetings," 2025. [Online]. Available: https://forwork.meta.com/horizon-workrooms/. [Accessed: 2025-04-27].

[4] Statista, "Barriers to mass consumer adoption of VR," 2019. [Online]. Available: https://www.statista.com/statistics/1099109/barriers-to-mass-consumer-adoption-of-vr/. [Accessed: 2025-04-29].

[5] "Zoom," 2025. [Online]. Available: https://www.zoom.com/. [Accessed: 2025-04-27].

[6] "Microsoft Teams," 2025. [Online]. Available: https://www.microsoft.com/sk-sk/microsoft-teams/group-chat-software. [Accessed: 2025-04-27].

[7] "Google Meet," 2025. [Online]. Available: https://meet.google.com/landing. [Accessed: 2025-04-27].

[8] G. Fauville, "Zoom Exhaustion & Fatigue Scale," 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2451958821000671 . [Accessed: 2025-04-29].

[9] Zaman, Faisal & Anslow, Craig & Chalmers, Andrew & Rhee, Taehyun. (2023). MRMAC: Mixed Reality Multi-user Asymmetric Collaboration. 10.1109/ISMAR59233.2023.00074.

[10] "Mirror Networking," 2025. [Online]. Available: https://mirror-networking.com/. [Accessed: 2025-04-27].

[11] H. Lyu, C. Vachha, Q. Chen, O. Pyrinis, A. Liou, B. T. Kumaravel, and B. Hartmann, "WebTransceiVR: Asymmetrical Communication Between Multiple VR and Non-VR Users Online," in *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, New York, NY, USA: Association for Computing Machinery, [1] 2022. DOI: 10.1145/3491101.3519816. [Online]. Available: https://doi.org/10.1145/3491101.3519816.

[12] "Rec Room," 2025. [Online]. Available: https://recroom.com/. [Accessed: 2025-04-27].

[13] "Unity," 2024. [Online]. Available: https://unity.com/. [Accessed: 2025-04-27].

[14] "XR Interaction Toolkit," 2025. [Online]. Available: https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@3.0/manual/index.html. [Accessed: 2025-04-27].

[15] "Agora.io," 2025. [Online]. Available: https://www.agora.io/en/. [Accessed: 2025-04-27]. [15] "Photon," 2025. [Online]. Available: https://www.photonengine.com/pun. [Accessed: 2025-04-27]

[16] Agora, "SD-RTN™ Delivers Real-Time Internet Advantages," White Paper, 2021. [Online]. Available: https://hello.agora.io/rs/096-LBH-766/images/Agora_WP_SD-RTN-Delivers-RealTime-Internet-Advantages.pdf.