Addressing Scalability for Real-time Multiuser Holo-portation: Introducing and Assessing a Multipoint Control Unit (MCU) for Volumetric Video

Sergi Fernández
i2CAT Foundation
Universitat Politècnica de Catalunya
Barcelona (Spain)
sergi.fernandez@i2cat.net

Mario Montagud
i2CAT Foundation
Universitat de València
Barcelona (Spain)
mario.montagud@i2cat.net

David Rincón
Universitat Politècnica de Catalunya
Barcelona (Spain)
david.rincon@upc.edu

Jaume Moragues
i2CAT Foundation
Barcelona (Spain)
jaume.moragues@i2cat.net

Gianluca Cernigliaro
i2CAT Foundation
Barcelona (Spain)
gianluca.cernigliaro@i2cat.net

ABSTRACT

Scalability, interoperability, and cost efficiency are key remaining challenges to successfully providing real-time holo-portation (and Metaverse-like) services. This paper, for the first time, presents the design and integration of a Multipoint Control Unit (MCU) in a pioneering real-time holo-portation platform, supporting realistic and volumetric user representations (i.e., 3D holograms), with the aim of overcoming such challenges. The feasibility and implications of adopting such an MCU, in comparison with state-of-the-art architectural approaches, are assessed through experimentation in two different deployment setups, by iteratively increasing the number of concurrent users in shared sessions. The obtained results are promising, as it is empirically proved that the newly adopted stream multiplexing together with the novel per-client and per-frame Volumetric Video (VV) processing optimization features provided by the MCU allow increasing the number of concurrent users, while: (i) significantly reducing resources consumption metrics (e.g., CPU, GPU, bandwidth) and frame rate degradation on the client side; and (ii) keeping the end-to-end latency within acceptable limits.

CCS CONCEPTS

• Computer systems organization → Real-time systems
• Networks → Network Services

KEYWORDS

Cloud Computing, Holo-portation, Multipoint Control Unit (MCU), Social VR, Virtual Reality (VR), Volumetric Video

ACM Reference format:
Sergi Fernández, Mario Montagud, David Rincón, Jaume Moragues, & Gianluca Cernigliaro. 2023. Addressing Scalability for Real-time Multiuser Holo-portation: Introducing and Assessing a Multipoint Control Unit (MCU) for Volumetric Video. In *Proceedings of ACM Multimedia Conference (MM'23), Oct. 29-Nov. 3, Ottawa, ON, Canada. ACM, New York, NY, USA, 9* pages. https://doi.org/10.1145/3581783.3613777

1 INTRODUCTION

Social Virtual Reality (VR) has become a promising *medium* to overcome key limitations of traditional 2D videoconferencing services, providing remarkable benefits in terms of quality of interaction, social meaning and (co-)presence [1, 2]. The user representation in Social VR becomes a key factor contributing to such benefits. Currently, most existing Social VR (or Metaverse) platforms are based on adopting (either cartoon-like or human-like) synthetic avatar-based user representations [2, 3]. However, recent studies have provided initial evidence on the benefits of providing realistic and volumetric holographic representations in such environments, not only to enhance the mediated experience but also the feeling of embodiment [1, 4], thus motivating the need for further research in this area.

In this context, recent works have shown the feasibility of providing realistic and Volumetric Video (VV) user representations, using low-cost off-the-shelf sensors, either as meshes (e.g., [1, 3]) or Point Clouds (e.g., [4, 5]). However, the integration of VV user representations in real-time multi-user holo-portation (i.e., holographic tele-transportation) scenarios still faces many challenges in terms of resolution, scalability, and adaptability, among others, thus limiting both the number and types of clients that can concurrently participate in shared virtual experiences (e.g., [3, 4]). On the one hand, research efforts have been devoted to proposing advanced client-driven Point Cloud compression techniques (e.g., [6, 7]) and viewport-aware processing and delivery strategies (e.g., [8, 9]) to improve the real-time performance and quality adaptation in VV services, like Point Cloud streaming. On the other hand, recent research efforts have been devoted to preliminarily exploring network-based media processing strategies to overcome scalability and adaptability limitations in holo-portation services, by means of proposing in-cloud Multipoint Control Unit (MCU) components (e.g., [10, 11]). While acknowledging the relevance of such contributions, further research is still needed to determine the feasibility and implications of integrating MCU components in multiuser holo-portation services in real scenarios, when exploiting key intrinsic characteristics from the 3D VV world, like dynamic relative distances and viewports for each user.

This work fills this gap, representing a unique contribution and a significant step forward compared to state-of-the-art contributions in this relevant research area. On the one hand, for the first time, a full service MCU has been designed and integrated into a pioneering end-to-end platform to enable real-time multi-user holo-portation services, based on VV user representations (using Point Clouds). Apart from stream multiplexing, the proposed MCU implements novel per-client and per-frame VV optimization strategies in the 3D world, like scene fusion, Level of Detail (LoD) adjustment, and data transmission, based on relative distances and viewpoints, thus being able to provide single and optimized streams to each involved client. On the other hand, the feasibility and implications of adopting such an MCU, in comparison with state-of-the-art architectural approaches based on the use of basic stream Forwarding Units (FU) (e.g., used in [1, 3, 4]), are assessed through experimentation in two real deployment setups, one in (constrained) distributed lab facilities, and one on an Amazon Web Service (AWS) cloud infrastructure. The obtained results, yet not conclusive and generalizable, prove that the addition of the VV MCU allows increasing the number of concurrent users in shared sessions, while keeping the latency limits within acceptable limits, providing smooth multiuser communication sessions, and significantly reducing the bandwidth and computational requirements on the client side.

Therefore, the contributions of this work determine that the usage of in-cloud VV processing components, like an MCU, is feasible in real multiuser holo-portation scenarios and contribute to providing higher scalable sessions and using more lightweight clients, which are essential requirements for next-generation Metaverse services. In essence, the presented technological contributions open the door to further research in this area, in terms of novel technological contributions (e.g., VV codecs, adaptive low-latency methods), deriving Quality of Experience (QoE) centric adaptation strategies, exploiting virtualization and cloud continuum paradigms, and conducting related experiments.

2 RELATED WORK

This section firstly introduces state-of-the-art architectural and communication models for multiuser conferencing services. Then, it briefly reviews the evolution from videoconferencing to Social VR platforms. Finally, it presents recent works in the scope of VV-based Social VR or holo-portation scenarios.

2.1 Architectural approaches for conferencing

Multi-party videoconferencing systems exist since almost three decades ago, and both the research community and industry have devoted major efforts to optimizing their performance in terms of key factors, like delays (e.g., [12]), video quality (e.g., [13]), and scalability (e.g., [14]), while trying to maximize the resulting QoE. Depending on the target requirements, three main architectural / communication approaches can be adopted in this domain [15]:

- Peer-to-peer: each client sends and receives one stream to / from all other clients, forming a mesh without any server involved in the data exchange process. It minimizes latency since the streams are directly exchanged between clients, but it presents scalability problems if the number of clients goes up.
- Client-server with a central router, typically known as FU: each client sends one stream, which is then replicated and sent to the other clients by the FU. The FU is also commonly known as Selective Forwarding Unit (SFU) if it performs selective forwarding strategies.
- Client-server with a central processing bridge, typically known as MCU: each client sends one stream and receives one fused stream from all other clients. It maximizes scalability and minimizes the total number of streams and usage of resources on client side, although it requires high computing and bandwidth processing resources on the server side, and typically has an impact on the resulting delays. The usage of (S)FUs typically provides better balance in terms of computational efforts between clients and server than the usage of MCUs, although each client has to process almost as many streams as clients in the session.

2.2 Social VR and holo-portation

Many Social VR platforms are recently appearing to palliate limitations of 2D videoconferencing services and exploit opportunities due to the emerging demands and situations, like lockdown, virtual events, entertainment, wide adoption of tele-work, etc. However, most Social VR platforms to date are based on representing the users as synthetic avatars [3], which still results in key limitations in terms of self-embodiment, identity preservation, recreating body language and social cues, among others, as e.g. reported in [1, 2, 16]. In this context, recent studies (e.g., [1, 17]) have provided initial empirical evidence on two key aspects: (i) state-of-the-art avatar-based Social VR platforms provide better quality of interaction, social meaning and (co-)presence levels than 2D videoconferencing platforms; and (ii) these levels are further increased if using realistic users' VV representations in these experiences, even resembling face-to-face settings. Such findings are promising, especially considering that the visual resolution of VV-based holograms in the adopted Social VR platforms in these works [1, 4]) have a significant margin of improvement. Similarly, recent studies have remarked the readiness of Social VR platforms to provide successful shared experiences between remote people, like meetings and virtual events, as surveyed in [2]. Interestingly, the work in [4] not only proved the appropriateness of Social VR, adopting state-of-the-art VV techniques (e.g., [5], [6]) and communications via an FU and Websocket-based protocols, to perform interactive virtual meetings, but also identified performance bottlenecks on the client side when increasing the number of users per session from 4 onwards (with Point Cloud streams at 15 frames per second (fps) and ~50K points per frame), even when using client PCs equipped with Graphics Processing Unit (GPUs).

In this context, two recent works have aimed at improving the scalability and adaptability of real-time holo-portation services. On the one hand, the work in [11] proposed an MCU for color and depth (i.e., RGB-D) streams, relying on 2D video codecs for generating a mosaic of fused RGB-D streams and communications via Web Real-Time Communication (WebRTC), integrated in a web-based Social VR platform. The usage of the MCU was

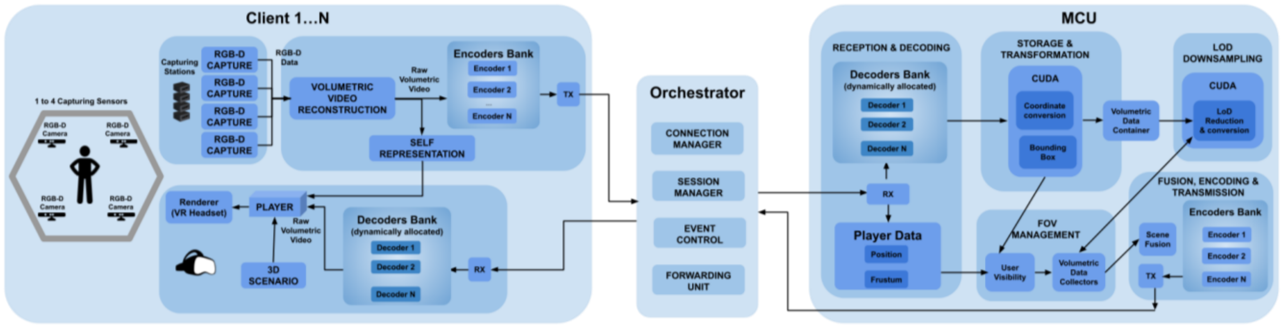

Figure 1: High-level system architecture of the holo-portation platform integrating the presented VV MCU

evaluated in a scenario with simulated users, showing that the proposed system can support sessions with up to 8 concurrent users, while reducing the processing requirements on the client side compared to the adoption of a FU-based session. However, the impact on delays and on session fluidity was not assessed. In addition, although the proposed MCU for holo-portation can leverage web-compliant and GPU friendly 2D video codecs, it neither supports fully VV representations (i.e., a single RGB-D sensor is used to capture the frontal view of the users) nor exploits the intrinsic characteristics and opportunities of using MCUs in a 3D environment, like relative distances and viewpoints. On the other hand, the work in [10] was pioneering in introducing the MCU concept for fully volumetric Point Cloud (i.e. VV) streams and 3D virtual environments, including simple viewport-aware and LoD adaptation strategies for fused scenes including all incoming VV streams from a shared session. However, that work contributed with just a preliminary proof-of-concept of an MCU able to process (two) pre-recorded holograms in a simple scenario, just providing initial insights of its potential. That MCU was not deployed and integrated in any real-time multi-user holo-portation platform and was not thoroughly evaluated in terms of scalability and other key metrics, like latency and fps variability.

Thus, the work presented in this paper represents a unique contribution and a significant step forward in this research area, as it is pioneering in: (i) evolving and deploying the concept of an MCU for VV, by exploiting key opportunities in the 3D domain, and being able to provide, in real-time, single and optimized streams to each target client in a shared session; (ii) integrating the novel MCU into a state-of-the-art holo-portation platform based on VV user representations (e.g., [3, 4]); (iii) assessing the feasibility and implications of integrating such an MCU in relevant real scenarios when compared to a FU-based communication model, by iteratively increasing the number of users per session.

3 HOLO-PORTATION PLATFORM

This section describes the end-to-end holo-portation platform from [3], which has been significantly evolved and extended to integrate a novel full service and GPU-powered MCU for VV.

3.1 Architecture and key components

A high-level overview of the system architecture of the holo-portation platform is provided in Figure 1. Next, a brief description of its key components is provided (further details in [3]):

3.1.1 Capture and Reconstruction. Real-time Point Cloud capture and reconstruction sub-system [6], which can adopt setups from 1 to N off-the-shelf RGB-D sensors (e.g., Azure Kinect). The captured RGB-D streams are converted to Point Cloud frames and, if multiple capture sensors are used, they are then fused to provide a full volumetric body capture. The resulting VV is sent back to the local rendering engine to provide a self-representation of the local participant, and sent to the encoding engines, which support a dynamically allocation of encoders, for its delivery to the remote participants.

3.1.2 Compression and Transmission. On the one hand, the platform adopts the Point Cloud codec from [5, 6], which exploits octree occupancy to represent the geometry and projects the colors onto a 2D grid. In the presented experiments, just I frames are used to favor real-time performance at the cost of compression efficiency. On the other hand, *socket.io*[1] (library built on top of WebSocket protocol) is used for exchanging data between the involved components, including the VV and audio streams.

3.1.3 Client. It has been developed in Unity (Windows build) and integrates the different sub-systems and modules for VV capture, reconstruction, encoding, decoding, transmission, reception, presentation of media content, and interaction modalities. It also provides the appropriate user interfaces for session management and enabling diverse interactions.

3.1.4 Orchestrator. It provides session and stream management features to enable multi-user holo-portation services, by interfacing a FU component for VV streams exchanges. In this work, the Orchestrator from [3] has been evolved to interface the presented VV MCU, by providing a session identifier to the MCU, so that the latter can join the session to perform in-cloud processing tasks and provide single output streams to the clients via the FU (processes detailed in Section 3.2).

3.2 MCU

The VV MCU is an in-cloud (virtualizable) component that receives all VV (Point Cloud) streams from the clients via the FU module (interfaced with the Orchestrator) and, after key processing tasks, sends back a single and personalized stream to each client, also via the FU. The MCU builds a scene with the VV data, geometry, positions, and viewpoints from each involved

[1] Socket.io: https://socket.io/ Last accessed in August 2023.

client, and it additionally performs viewport-aware processing and LoD adaptation strategies to deliver a single optimized VV stream to each client. As payback, the MCU needs to receive, decode, fuse, process, re-encode and transmit the transformed VV streams, resulting in extra computational load and latency.

To ease its integration in the end-to-end platform, the MCU acts as an additional client interfacing with the Orchestrator for data exchange. However, unlike clients, the MCU just sends back to the FU/Orchestrator a unique optimized stream to each target client in the shared session.

The MCU is composed of 5 main sub-components with associated modules (Figure 1):

3.2.1 Reception and Decoding. After connecting with the Orchestrator, the VV streams for each client are received and provided to a bank of decoders running in CPU. This decoder bank can be dynamically and strategically allocated and parametrized, based on the VV stream properties and their number. Once uncompressed, a set of operations are executed in GPU to transform the geometric data. These are major advantages compared to the proof-of-concept MCU from [10].

3.2.2 Volumetric Data Transformation and Storage. The VV frames from each client are represented in their own coordinate system defined during the calibration of the capture sub-systems. Thus, all voxels need to be transformed, by exploiting GPU capabilities, to a common scene-referenced coordinate system to provide a single fused geometry. After that, the fused geometry is stored in RAM in a Volumetric Data Container, and the associated bounding box is calculated (in GPU) prior performing viewport-aware and LoD adjustment strategies.

3.2.3 Field-of-View (FoV) Management. When multiple users are integrated within a shared 3D virtual environment, their positions, relative distances between them, and individual viewpoints, become paramount factors to optimize the amount of data to be transmitted. As a first example, the resolution for the holograms of a close-by user needs to high, but the one of a further away user can be downgraded without potentially having a negative impact on the perceived QoE. As a second example, the resolutions of holograms in the Main (Central) Vision Area need to be high, but the ones of those in peripheral areas could be downgraded, and even the ones for non-visible users could be omitted. Based on these assumptions and optimization rules, a FoV Manager has been designed and adopted to inspect the positions of VV elements in the scene, according to the position and viewpoint of each target client, and to perform strategic LoD adjustment methods to be applied in the posterior fusion and encoding processes for each client.

3.2.4 LoD Downsampling. The calculation and adjustment processes of the LoD levels for VV elements in a scene, depending on the target user's FoV, relative distances and viewpoints, and potentially other contextual factors, become a research topic that needs to be further investigated and validated via QoE testing. This work has adopted a simple, yet smooth and adaptative, algorithm to derive the LoD levels for voxels of a VV representation, V_r, as a starting point to prove the feasibility and impact of such a feature. In this algorithm, V_r is calculated by considering adaptation weights associated to viewpoints, w_f, and distances, w_d, and applying associated linear coefficients c_f and c_d (where $c_f + c_d = 1$):

$$V_r = w_f \times c_f + w_d \times c_d \quad (1)$$

Based on V_r calculation, the LoD Optimization module takes the geometry from the Volumetric Data Container module and pushes the data to the GPU to optimize the resulting geometry with downgraded LoD levels.

3.2.5 Fusion, Encoding and Transmission. The fused scenes for each client are constructed by concatenating the geometries obtained from the Volumetric Data Collector module. Then, a bank of encoders is dynamically allocated to compress the sequence of such scenes. Finally, the resulting encoded VV frames are transmitted to the target clients via the FU using socket.io.

4 EVALUATION

This section aims at analyzing the implications of integrating the VV MCU in a real multiuser holo-portation scenario, compared to the use of just a FU, which has been the architectural approach adopted in the existing real-time multiuser VV services up to date (e.g., [1, 3, 4]). The evaluation is focused on assessing the impact of the stream multiplexing, LoD Selection and FoV removal features provided by the VV MCU, in scenarios with varying distances between users and viewpoints, in terms of key resources consumption metrics (e.g., CPU, GPU, bandwidth) [18] and performance metrics (e.g., fps, end-to-end latency…) through the execution of two experiments. Both experiments use the same user layout, but they mostly differ in the underlying infrastructure (infra) and scenario for deploying the platform components and connecting them.

Figure 2 sketches the configured users' layout in the experiments, in a V shape, where a target client can receive up to N (with N varying within [2, 4, 6, 8]) VV streams representing users integrated in the virtual environment. The virtual users' positions for each of the lines of the V shape are progressively separated in steps of 45 degrees, and the (virtual) distances between each of the lines and the one for the target user are set to 5m, 10m, 20m and 30m. Such a layout is meant to accomplish three conditions: (i) all the users are inside the target user's FoV when its virtual camera is directed to the frontal position (i.e., 0°); (ii) the resulting LoD for the different user representations significantly vary due to relative distances and viewpoints in the scenario; and (iii) the effect of FoV removal can be observed if the target user gradually changes the viewpoint.

Although the MCU has been tested and validated in multiuser holo-portation sessions with the integration of real users captured in real-time, all participants in the conducted experiments are represented as pre-recorded VVs (i.e., Point Clouds) by using the Loot sequence from the 8i dataset [19]. These sequences have been downsampled to 15 fps and ~50k points per frame to resemble a holographic communication scenario with real-time captured users, according to the VV stream parameters and traces from [3, 4].

The following settings were applied to the MCU parameters (Section 3.2.4) in the conducted experiments: (i) bank of encoders set to 9; (ii) bank of decoders set to 24; and (iii) w_f set to 0.5 or 1, depending on whether the user is in the peripheral or main vision area, w_d defined as a sigmoid curve (adopting values from 1 to 0) depending on the relative distance, and both c_f and c_d set to 0.5. In each experiment, for each iteration (N=2, 4, 6, 8) and architectural approach, the tests are repeated 10 times.

4.1 Exp. 1: Lab infra and pruned MCU

Experiment 1 (Exp. 1) aims to compare the FU (i.e., no MCU) and MCU test conditions when deploying the platform in the available (constrained) distributed lab facilities, and when considering an ideal scenario, i.e. a setup in which the MCU has only to provide a personalized stream to the target user to avoid the server running it to become overloaded, and thus having an impact on the ideal output stream to be provided by it.

4.1.1 Setup and Methodology. For Exp.1, the involved clients in each test condition have been equally distributed in two remote networks in a metropolitan area, with an effective bandwidth capacity of 100 Mbps. It should be noted that the scenario in this experiment assumes that the MCU just delivers a single output stream to the client of the target user, which can be a realistic case when all other users do not have any user in their FoV. This configuration has been applied to ease the isolation of problems and replication of experiments, and to ensure that the available computing and network resources perform satisfactorily in not so restricting test conditions, without getting saturated before expected.

4.1.2 Results. This subsection reports on the obtained results for the considered metrics (average values from 10 repetitions) for each test condition varying the number of users (N=2, 4, 6, 8) and architectural approach (no MCU vs MCU), whose temporal evolution is depicted in all graphs collected in Figure 3.

Exp. 1 was initially planned to iterate with an incremental number of received VV representations until 8 per session plus the target user (Figure 2). However, as it can be observed in the graphs in Figure 3, in the FU mode, the platform started to perform in an unstable manner in test conditions with N=6 users (crashing in 5 out of 10 iterations), while it always crashed when a 7th user was activated. That is the reason why the results are provided for N=7 users, and not for N=8, as it is the higher number of users for which comparable results can be provided.

Crashes can be identified in the graphs with drastic reductions in the metrics' values for test conditions with 6 and 7 users (although the platform was still able to keep the service running for the active clients).

For stable sessions (i.e., with 2 and 4 users), the graphs evidence that both architectural approaches behave as expected. While in the FU mode the resources consumption metrics hardly fluctuate throughout the duration of the session, in the MCU mode their fluctuation is more pronounced, as the amount of data delivered to the target client depends on the instantaneous relative distances and viewpoints in such a case.

In order to automatize viewpoint movements and ease replicability, a specific pattern (Figure 2) for the target user camera view has been setup, including movements from left (-100°) to right (100°), waiting 20 seconds when each user position enters and leaves the FoV, resulting in a test duration around 5 minutes.

Table 1 provides the specifications of the computing machines in which each platform component has been deployed.

To support this, a yellow graph has been added in these figures, which represents the fusion size (i.e. the number of points per fused frame after applying LoD reduction strategies) in the analyzed test condition. In general, the graphs in Figure 3 evidence that the MCU mode results in a reduction of the computation load (CPU and GPU) and bandwidth consumption compared to the FU mode, in any comparable test condition.

Regarding the end-to-end latency, the FU mode performs better than the MCU mode. In sessions with 2 and 4 users, the MCU introduces an extra delay between 100ms and 200ms, reaching latency levels around 300ms. In test conditions with 6 and 7 users, the delays reach levels between 500ms and 700ms, but are kept quite stable and still within allowable limits for real-time services, as reported in [20]. In such cases, the MCU mode avoids the overload of the target client, which does not happen in the FU mode. Finally, both the average and standard deviation values of the fps on the client side degrade in a more pronounced manner in the FU mode than in the MCU mode, which may negatively impact the perceived QoE (e.g., [21]). Thus, the results for these key metrics (latency and fps) evidence the benefits provided by the MCU when the number of users goes up.

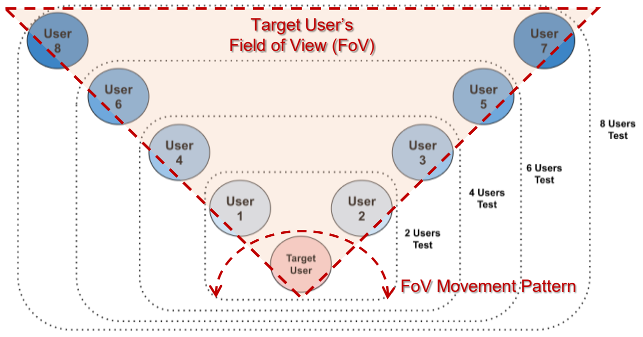

Figure 2: Users' layout in the conducted experiments

Target Client	CPU: Intel Core i9-12900H @ 3.20GHz RAM: 32 GB @ 2133MHz GPU: NVIDIA GeForce RTX 3070 Ethernet Card: Intel(R) Ethernet Controller (3) I225-V
MCU	CPU: Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz RAM: 192 GB @2666 MHz GPU: NVIDIA Tesla T4 Ethernet Card: Cisco X550-TX 10 Gig LOM
Orchestrator	CPU: Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz RAM: 192 GB @2666 MHz GPU: NVIDIA Tesla T4 Ethernet Card: Cisco X550-TX 10 Gig LOM

Table 1: Hardware specifications in Exp. 1

4.2 Exp. 2: Cloud infra and full service MCU

Exp. 2 aims to compare the FU and MCU test conditions in a less constrained deployment scenario, leveraging on cloud infra to allocate both the server and client components, and on higher-capacity and more stable network conditions. This is meant to avoid early bottlenecks in terms of computational and network resources when running the MCU in multiuser sessions, and to ease replicability of the experimentation.

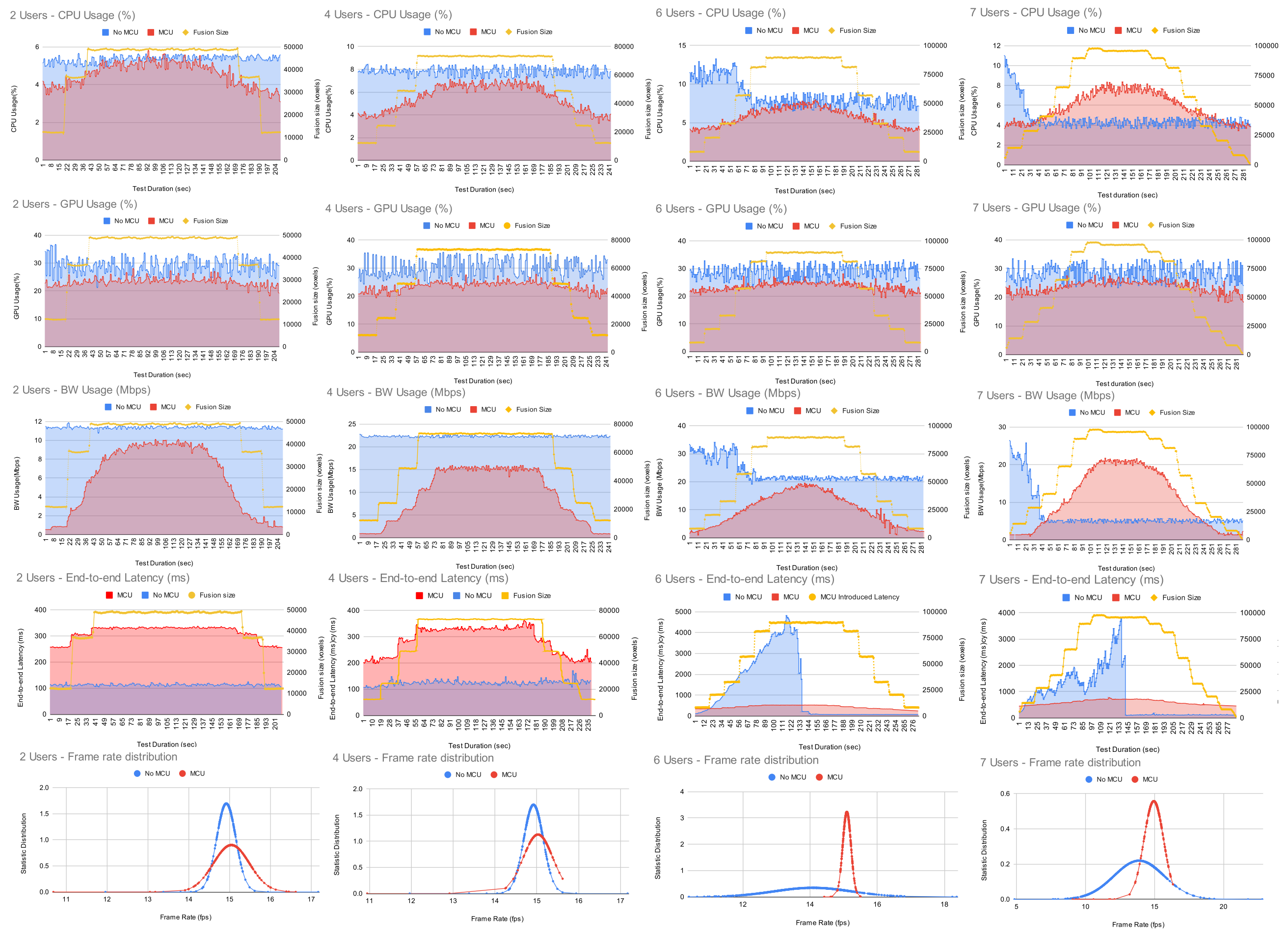

Figure 3: Exp. 1 results (CPU, GPU, bandwidth, fps)

In addition, two further conditions have been enabled to resemble more realistic and challenging situations: (i) the ideal server-side scenario from Exp. 1 has been dismissed, but now a "full service" MCU has been enabled, meaning that the MCU provides a personalized output VV stream for each user in the session; (ii) the relative distances between users and their viewpoint patterns have been slightly modified to more accurately resemble realistic scenarios and intensify the processing needs for the MCU.

4.2.1 Setup and Methodology. Exp. 2 follows the same approach and methodology than Exp. 1, but four main modifications have been applied to overcome identified limitations, reflect more realistic and challenging scenarios, and ease replicability of the experimentation. First, all involved platform components (Orchestrator, MCU, and clients) have been deployed on an AWS cloud infra. In particular, the clients have been deployed on an AWS EC2 G4 instance, the MCU on an AWS EC2 G5 instance, and the Orchestrator on an AWS EC2 T3 instance (specifications in Table 2). If these specifications are compared to the ones from Exp. 1, it can be observed that the clients have now lower computing capacity, as it was proven to not being a limiting factor in Exp. 1, and the MCU has now higher capacity, as a full service MCU is to be activated. A second improvement resides in the available network resources, as 10 Gbps full duplex connections among all deployed components are available in this AWS setup. This is meant to avoid early congestion situations, and to support in validating that crashes in Exp. 1 were not due to processing but to network limitations. Third, the relative distances between the lines of the V shape (Figure 2) have now been setup to 5m, so that users in general get closer to the target user, and thus less aggressive LoD adjustment levels are performed. Finally, all users' point of views now rotate from -90° to 90° iteratively at a pace of 1° per second. These movement patterns ensure that all users have at some point other users in their FoV.

Clients	CPU: 16 vCPU AMD EPYC 7R32 2,80 Ghz; RAM: 64 GB GPU: AMD Radeon Pro V520 BW available: 10 Gbps full duplex
MCU	CPU: 64 vCPU AMD EPYC 7R32 2,80 Ghz; RAM: 256 GB GPU: NVIDIA A10G Tensor Core BW available: 10 Gbps full duplex
Orchestrator	CPU: 4 vCPU Intel Skylake E5 2686 v5 3 Ghz; RAM: 64 GB GPU: not included. BW available: 10 Gbps full duplex

Table 2: Hardware specifications in Exp. 2

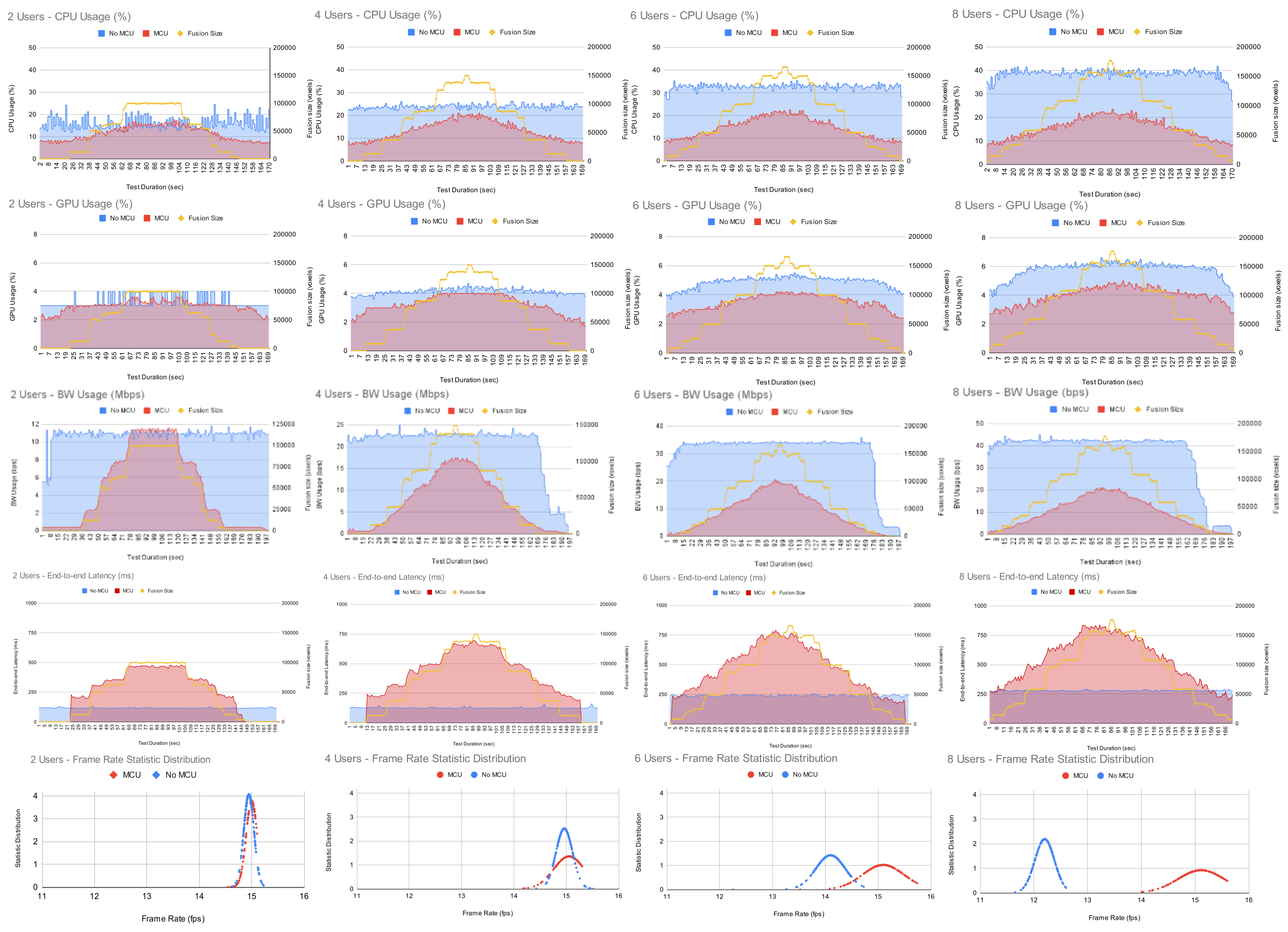

Figure 4: Exp. 2 results (CPU, GPU, bandwidth, fps)

4.2.2 Results. As for Exp. 1, this subsection reports on the obtained results for the considered metrics, varying the number of users (N=2, 4, 6, 8) and for each architectural approach, whose temporal evolution is sketched in all graphs collected in Figure 4.

A general insight and remark for Exp. 2 is that, despite reducing the computational resources for the clients, no crashes have been observed for any test condition. This is particularly relevant since the VV stream properties are identical in both experiments, and even the relative distances between users have been shortened in this second experiment. This reinforces that crashes when using the FU mode in Exp. 1 were mostly due to the network capacity limitations rather than to processing limitations.

In general terms, the graphs in Figure 4 reflect, as for Exp. 1, a significant reduction of the computational and network resources usage when activating the MCU mode. It is also relevant to observe that the fusion size behaves differently in Exp. 2 compared to Exp. 1. On the one hand, its fluctuation is now determined by the newly adjusted viewport movement patterns to each of the clients. On the other hand, its size is almost double than in Exp. 1, since the users in the FoV of the target user are now closer, and thus a softer LoD downgrading strategy is applied. This in turn results in a slight increase of the bandwidth consumption in Exp. 2 compared to Exp. 1.

As for Exp. 1, the usage of the MCU results in an increase of the end-to-end latency, mainly due to the extra processing tasks. In the FU mode, the latency levels obtained in both Exp. 1 and 2 are very similar for comparable test conditions with 2 and 4 users. This evidences that no extra delay occurs by the fact of using a cloud infra. In the MCU mode, the latency levels obtained in Exp. 2 are a bit higher than in Exp. 1, but this was expected due to the higher fusion sizes and now having a full service MCU having to process the optimized VV streams for each user. Still the latency levels are kept below 800ms for any MCU test condition, which may not have a negative impact the perceived QoE [20]. Finally, the average and standard deviation values of the fps degrade in a more pronounced manner in the FU mode, which may negatively impact the QoE [21].

Thus, the obtained results for all these key metrics also evidence the benefits provided by the MCU when the number of users goes up, even when having a full service MCU.

5 DISCUSSION, CONCLUSIONS AND FUTURE WORK

Multi-user holo-portation systems are attracting the interest of the research community and industry alike. While major research efforts are being devoted to maximizing the visual quality, to providing adequate streaming protocols, and to enabling adequate multi-modal interaction features, scalability, interoperability, and cost efficiency remain unsolved challenges in this research area.

For the first time, this paper has reported on a full service MCU for VV integrated in a holo-portation platform [3, 4], supporting realistic and volumetric user representations, to increase its scalability and adaptability. The feasibility and implications of integrating the VV MCU in real-time multi-user holo-portation services have been assessed by performing experiments in realistic scenarios, by iteratively increasing the number of users per session when adopting FU-based and MCU-based architectural approaches. The obtained results prove that the addition of the MCU can significantly reduce the resources consumption and fps degradation on the client side, while keeping the latency limits within stable and acceptable limits for real-time communication services.

The obtained results, although insightful, should be considered neither generalizable nor conclusive, due to the specific characteristics of the evaluated scenarios and test conditions, and to other identified limitations regarding the conducted experiments and the presented MCU implementation, which are discussed next.

First, a notable limitation of Exp. 1 is the limited capacity of the distributed networked scenario in which it has been conducted. Second, Exp. 1 has been based on just providing a single personalized VV stream for the target user. These two limitations have been addressed in Exp. 2. Third, the tested virtual scenario and configured viewport patterns are very similar in both experiments, but not identical. On the one hand, this has an impact on the accuracy of the comparability of results but, on the other hand, also helps in assessing the impact when modifying some relevant settings, like the distances between users and the width and velocity of viewing patterns.

Thus, it is considered an acceptable limitation that helps in shedding some light on the relevance of such aspects. Fourth, both experiments have been conducted for the same number of users, but the test conditions in Exp. 2 did not reach the upper limits for the clients and MCU capacity. Further studies should be conducted to provide an evaluation framework to compare between test conditions more accurately, and to determine scalability limits, ideally being able to extrapolate these limits to any desired scenario. Anyway, the conducted experiments are considered insightful enough to prove the benefits of the MCU, and its satisfactory performance in scenarios with higher number of users than in any multiuser VV or holo-portation service / study reported in literature.

Furthermore, limitations regarding the current MCU architecture and to a subset of its implemented features and methods have been identified. First, although being a full service MCU, it is (still) not ready to horizontally scale up, by providing efficient intra- and inter-session resources management. Second, the current version of the MCU does not implement any caching strategy, which would allow reusing geometry calculations, LoD transformations, and encoded (sub-)scenes for different target clients (e.g., close users in the virtual world).

As a third limitation, neither the Orchestrator nor the MCU integrate any smart strategy to adapt the streams to be provided to the clients based on their capabilities, current activity or roles. Finally, although the usage of WebSocket protocol (socket.io) has been proven to perform satisfactorily, the adoption of more advanced and adaptive streaming protocols (e.g., WebRTC) should be assessed.

In addition, despite that the platform has been tested with real users, on the one hand the presented experiments have been conducted with pre-recorded holograms and, on the other hand, subjective tests would need to be conducted to find out the most appropriate strategies and values for the associated methods and parameters, especially when it comes to viewport-aware processing and LoD adjustment, and to determine the impact on the QoE, in a variety of representative scenarios.

Future work will be targeted at overcoming the discussed limitations and at evolving the associated MCU components , as well as at enabling their smart deployment over the cloud continuum in next-generation networks.

Finally, it should be remarked that despite a wide set of limitations have been identified, the contributions and lessons learned from this paper are not only insightful and significant, but interestingly open the door to further remaining research opportunities in this area, which can be explored thanks to the development of pioneering technological contributions and testbeds for a communication, interaction, and collaboration *medium* expected to become dominant in the near future.

ACKNOWLEDGMENTS

This work has been funded by MCIN/AEI/10.13039/501100011033 / FEDER, with Grant PID2021-126551OB-C21 (HoloMIT 2.0 project), and by the Spanish Ministry of Economic Affairs and Digital Transformation and the European Union – NextGeneration EU, in the framework of the Recovery Plan, Transformation and Resilience (PRTR) (Call UNICO I+D 5G 2021, Ref. TSI-063000-2021-4 – 6G-Openverso-Holo). In addition, this work has been supported by the Horizon Europe (Smart Networks and Services Joint Undertaking) program, under agreement n° 101096838 (6G-XR project). The work of Mario Montagud has been funded by MCIN/AEI/10.13039/501100011033 under Grant RYC2020-030679-I and by "the European Social Fund (ESF) Investing in Your Future".

REFERENCES

[1] Mario Montagud, et al. 2022. Towards socialVR: evaluating a novel technology for watching videos together. Virtual Reality, 26, 1593–1613 (2022). https://doi.org/10.1007/s10055-022-00651-5
[2] Mario Montagud, et al. 2022. Social VR and multi-party holographic communications: Opportunities, Challenges and Impact in the Education and Training Sectors. arXiv:2210.00330
[3] Sergi Fernández, Mario Montagud, Gianluca Cernigliaro, David Rincón. 2022. Toward Hyper-Realistic and Interactive Social VR Experiences in Live TV Scenarios. IEEE Transactions on Broadcasting, vol. 68, no. 1, pp. 13-32, March 2022. doi: 10.1109/TBC.2021.3123499
[4] Sergi Fernández, Mario Montagud, Gianluca Cernigliaro, David Rincón. 2022. Multiparty Holomeetings: Toward a New Era of Low-Cost Volumetric Holographic Meetings in Virtual Reality. IEEE Access, vol. 10, pp. 81856-81876, 2022. doi: 10.1109/ACCESS.2022.3196285
[5] Jack Jansen, et al. 2020. A pipeline for multiparty volumetric video conferencing: transmission of point clouds over low latency DASH. ACM MMSys'20. Istanbul (Turkey), June 2020. https://doi.org/10.1145/3339825.3393578
[6] Rufael Mekuria, Kees Blom, Pablo César. 2017. Design, Implementation, and Evaluation of a Point Cloud Codec for Tele-Immersive Video. IEEE Transactions on Circuits and Systems for Video Technology, 27, 828-842, April 2017. doi: 10.1109/TCSVT.2016.2543039

[7] Sebastian Schwarz, et al. 2019. Emerging MPEG Standards for Point Cloud Compression. IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 9, no. 1, pp. 133-148, March 2019, doi: 10.1109/JETCAS.2018.2885981

[8] Jounsup Park, Philip A. Chou, Jenq-Neng Hwang. 2019. Rate-Utility Optimized Streaming of Volumetric Media for Augmented Reality. IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 9, no. 1, pp. 149-162, March 2019, doi: 10.1109/JETCAS.2019.2898622

[9] Shishir Subramanyam, Irene Viola, Alan Hanjalic, Pablo César. 2020. User Centered Adaptive Streaming of Dynamic Point Clouds with Low Complexity Tiling. ACM International Conference on Multimedia (MM'20), Seattle (US), October 2020

[10] Gianluca Cernigliaro, et al. 2020. PC-MCU: point cloud multipoint control unit for multi-user holoconferencing systems. 30th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSDAV'20). Istanbul (Turkey), June 2020

[11] Simon Gunkel, et al. 2021. VRComm: an end-to-end web system for real-time photorealistic social VR communication. 12th ACM Multimedia Systems Conference (MMSys'21). Istanbul (Turkey), June 2021

[12] Mohammad H. Hajiesmaili, et al. 2017. Cost-Effective Low-Delay Design for Multiparty Cloud Video Conferencing. IEEE Transactions on Multimedia, vol. 19, no. 12, pp. 2760-2774, Dec. 2017, doi: 10.1109/TMM.2017.2710799

[13] Marwin Schmitt, Judith Redi, Dick Bulterman, Pablo Cesar, P. 2018. Towards Individual QoE for Multiparty Videoconferencing. IEEE Transactions on Multimedia, vol. 20, no. 7, pp. 1781-1795, July 2018, doi: 10.1109/TMM.2017.2777466

[14] Yu Wu, Chuan Wu, Bo Li, and Francis C.M. Lau. 2013. vSkyConf: cloud-assisted multi-party mobile video conferencing. ACM SIGCOMM Workshop on Mobile cloud computing (MCC'13). 33-38. Hong Kong (China), August 2013, https://doi.org/10.1145/2491266.2491273

[15] Dunja Vučić, Lea Skorin-Kapov, Mirko Suznjevic. 2016. The impact of bandwidth limitations and video resolution size on QoE for WebRTC-based mobile multi-party video conferencing. screen, 18, p.19, 2016

[16] Chiwon Lee, Hyunjong Joo, Soojin Jun. 2021. 2021. Social VR as the New Normal? Understanding User Interactions for the Business Arena. ACM CHI EA'21. Yokohama (Japan), May 2021, https://doi.org/10.1145/3411763.3451740

[17] Jie Li, et al. 2019. Measuring and Understanding Photo Sharing Experiences in Social Virtual Reality. ACM Conference on Human Factors in Computing Systems (CHI'19). Glasglow (UK). May 2019, https://doi.org/10.1145/3290605.3300897

[18] Mario Montagud, et al. 2020. Open-source software tools for measuring resources consumption and DASH metrics. 11th ACM Multimedia Systems Conference (MMSys '20), Istanbul (Turkey), June 2020, https://doi.org/10.1145/3339825.3394931

[19] Eugene d'Eon, Bob Harrison, Taos Myers, Philip A. Chou. 2017. 8i voxelized full bodies-a voxelized point cloud dataset. ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006. 2017.

[20] Gunila Berndtsson, et al. 2018. Methods for Human-Centered Evaluation of MediaSync in Real-Time Communication. In: Montagud, M., et al. (eds) MediaSync: Handbook on Multimedia Synchronization. Springer, Cham. 2018, https://doi.org/10.1007/978-3-319-65840-7_9

[21] Mario Montagud, Fernando Boronat, Bernardino Roig, Almanzor Sapena. 2017. How to Perform AMP? Cubic Adjustments for Improving the QoE. Computer Communications. Volume 103, Pages 61-73, May 2017, https://doi.org/10.1016/j.comcom.2017.01.017