Relay-based network architectures for Collaborative Virtual Reality Applications

Fadia Hasnaoui[1], Lamia Zohra Mihoubi[1], Maria Pateraki[2,3] and Miloud Bagaa[4,5]
[1] ESI Alger, Algeria - e-mails: {gf_hasnaoui@esi.dz,gl_mihoubi@esi.dz}
[2] ORamaVR S.A., Heraklion, Greece - email: maria@oramavr.com
[3] National Technical University of Athens, Greece
[4] Aalto University, Espoo, Finland - email: miloud.bagaa@aalto.fi
[5] CSC-IT Center for Science Ltd. Espoo, Finland - email: miloud.bagaa@csc.fi.

***Abstract*—Currently deployed NAT devices are designed primarily around the client/server paradigm, in which relatively anonymous client machines inside a private network initiate connections to public servers with stable IP addresses and DNS names. Thus, the asymmetric addressing and connectivity regimes established by NAT devices have created unique problems for Peer-to-Peer (P2P) applications and protocols. Multiple NAT-traversal techniques have been developed to overcome these shortcomings, each offering a different set of pros and cons. In the context of a P2P collaborative virtual reality (CVR) system, the difficulty of selecting a convenient and effective NAT-traversal technique increases exponentially because of the added constraints related to CVR. In this view, this article discusses the trade-offs of different NAT-traversal techniques and the CVR challenges that need to be taken into account when choosing a NAT-traversal technique. Finally, it presents a relay-based approach that leverages container migration to mitigate the drawbacks that come with this solution and accentuate its advantages.**

I. Introduction

6G system is expected to target a broad range of services and applications rather than focusing only on regular consumers as seen in previous generations. According to [1], the ICT drivers towards the year 2030 aim to support the next frontier in applications and business cases in Virtual Reality (VR), Augmented Reality (AR), holographic-type communication services, digital twin, ubiquitous intelligence, tactile Internet, and multi-sense experience. These industrial verticals require rigorous requirements in terms of reliability, latency, throughput, and availability.

VR aims to immerse the user in a virtual world, providing the ability to interact naturally with the virtual environment (objects and virtual characters) mainly for training and entertainment purposes. With the recent interest in the paradigm of the metaverse and the expected thriving of the 6G, the social aspect in VR is increasingly supported in several application domains to realize collaborative VR experiences beyond the single user.

Collaborative Virtual Environments (CVE), a term synonymous with Collaborative Virtual Reality systems (CVR), are shared virtual spaces created to allow participants across a network to work together on common tasks while coexisting in the same virtual space. The collaborative activities can range from group work (designing systems, environment planning, product design, and modeling) to social activities and entertainment. CVR is currently one of the most challenging VR research areas. The single-user VR demands computational power to handle the incoming data from sensory input devices and render high-quality 3D graphics to simulate reality with high fidelity. The output is generated with a frame rate that is, at the very least, acceptable and comfortable to the human eye. Beyond the single-user scenario, CVR demands a high quality of service (QoS) to maintain multi-user virtual spaces' consistency. The network characteristics in terms of network bandwidth and latency are considered to ensure a coherent experience by all users, while for a smooth CVR experience with high quality of experience (QoE), the network should be optimized in terms of end-to-end delay, bandwidth, and reliability. Furthermore, to guarantee that users do not feel the difference between local and remote objects in the shared space, high interactivity and responsiveness are required.

In this view, this paper underscores the requirements of CVR systems and the network-related challenges that arise from these requirements and constraints and presents network architectures used in CVR systems and their limitations, followed by an explanation of how the introduction of relay services can fill these gaps.

The remaining of the paper is organized as follows, section II discusses relevant research in this field. Section III presents CVR, its requirements, and the resulting networking challenges. The proposed solution is described in section IV. Section V shows the experiment results of the proposed solution. Finally, the study is concluded in section VI with a critique of the suggested approach and an inspection of future research avenues.

II. Related Work

CVR network-related challenges and requirements have been covered in varying levels of depth and extensively. A taxonomy of Collaborative Virtual Environments (CVEs) has been given by Macedonia and Zyda in [2]. Authors in [3], [4] have discussed the challenges that VR systems have to overcome in order to fulfill untethered and interconnected VR's immense potential in manifold domains and the requirements of interconnected VR in terms of latency, reliability,

978-1-6654-3540-6/22

and bandwidth. Authors in [3] have presented a set of key enablers to achieve the future vision of an ultra-reliable and low latency VR by leveraging *mobile edge computing* and *proactive caching*. While the authors tackle the topic of network challenges in VR, they only skim over CVR and the added difficulties it brings, failing to address important requirements. The authors have identified network-related enablers for improving the performance of wireless head-mounted displays and exploiting an edge computer network. But, these suggested modifications are not always possible or the best option, especially in the case of P2P networks. Another important issue that was not considered is scalability. Huge overlay networks would arise as a consequence of edge computing, creating a problem for connectivity protocols [5], which are already an issue for P2P networks [6].

In [4], authors underscore the desperate need for a fundamentally new network architecture capable of satisfying VR's exceptionally high data rate and stringent latency requirements as VR rapidly goes beyond early use cases like games and moves towards social media trends and everyday life applications. The authors have identified key enablers of fully interconnected VR and promising research avenues. They conclude that a plethora of challenges in multiple domains needs to be addressed in order to achieve this future vision including caching/storage/memory, local/fog/edge/cloud computing, and processing, computer vision, and media and context-information and analytics. They focus on improving the VR system rather than the networking aspect, and short-range wireless communications that only decrease the latency between the headset and the PC running the CVR application rather than improving the quality of communication between the dispersed participants.

To conclude, the literature offers only a handful of tools and ideas to improve networking for VR and CVR. Even fewer products of literature approach the topic of NAT-traversal in the case of P2P CVR systems, let alone suggest the adaptation of the traditional NAT-traversal techniques to CVR. This paper comes to fill this gap by suggesting a comprehensive system that leverages relay service and NAT-traversal for enabling an efficient Collaborative Virtual Reality (CVR) environment.

III. Collaborative Virtual Reality (CVR)

CVEs are computer-generated spaces that support multiple participants across a network. The connected users are visually embodied in the shared environment as avatars. The users are autonomous entities, free to navigate independently through the terrain, encountering each other, artifacts, and data objects. However, they are also able to collaborate/work together on shared tasks and interact with each other via speech, text, or body gestures.

Virtual environments were first introduced as one user applications, but with the appearance of CVR, another layer of difficulties and challenges was added to the mix. The main concern for CVR systems is achieving a high level of realism for the connected users. In other terms, the users must have the feeling of being present in the same place and they must be able to see and react to others' actions when they happen. Thus, the manifold of technological challenges stemming from a variety of disciplines like computer networks, cloud computing, memory and storage, artificial intelligence, and computer vision needs to be addressed to achieve this vision.

This section presents the requirements and characteristics of CVR systems and focuses on the network-related challenges that need to be addressed in order to fulfill these requirements. In the balance of this section, we will present the network and CVR requirements and characteristics, then we present the network relay concept for enabling a collaborative virtual environment.

1) Bandwidth and network capacity: Supporting multiple users, video, audio, and real-time sharing of 3D graphics primitives and models necessitates a lot of bandwidth in distributed VR. In order to guarantee that all the users have the same current view/state on the shared 3D space when a user makes changes to the shared virtual environment, updates need to be sent to all the other users. The eye can capture 720 million pixels for each of the 2 eyes, 36 bits per pixel for full color and 60 frames per second, for a total of 3.1 trillion (tera) bits for some headsets, assuming no head or body rotation. Compression standards today can decrease that by a factor of 300 and even if future compression could hit a factor of 600, that still means that network throughput needs to be at least 5.2 gigabits per second [4].

2) Latency: Stringent latency criteria are of utmost importance in CVR environments in order to provide a pleasant immersive and interactive VR experience. A CVR application that allows virtual surgical training for medical students from around the world mandates that participants are made aware immediately of any action that another participant takes in the surgery room, otherwise the training is severely disrupted and inaccurate. For real-time systems like CVR, latency must be less than 100 ms and also less than the time needed to generate one graphics frame [7]. The fulfillment of the ideal CVR vision of a seamless immersive, interactive, and highly responsive virtual world that the geographically dispersed users can't distinguish from reality hinges on approaching this limit as much as possible. In order to reduce end-to-end latency to reasonable standards, it is important to reduce the communication delay [3].

3) Reliability: Reliability means that systems can logically assume that players are all able to connect successfully to the environment and send and receive data correctly. Users must receive update messages relating to changes in the state of a shared object in a reliable and timely manner. Reliability also includes failure management, which means the reaction of the system to a possible failure. For instance, if a user drops out from the network, the other users must continue their experience of the CVR application unaffected by the event. One of the biggest challenges of CVR is that both low latency and high reliability are stringent constraints of the system despite their conflicting nature. Systems requiring high-reliability mandate frequent re-sending of lost data packets, thus having longer network delays as re-sending packets

introduce lags in the system.

A. Relay

In P2P paradigm exists multiple NAT traversal solutions, namely port forwarding that may seem as a promising solution but presents security and privacy concerns since the user is opening ports and creating direct lines to the network for hackers and malware attacks. Furthermore, this solution requires the intervention of the average user which can be tricky and create some unnecessary complications. An alternative can be TCP/UDP Hole Punching which requires all intermediary routers to be EIM-NATs (Endpoint-Independent Mapping Network Address Translators). However, a significant percentage of deployed NATs use Endpoint-Dependent Mapping and do not allow TCP/UDP hole punching. But, the use of a relay, to implement a sort of hybrid peer-to-peer with client/server architecture is quite promising. This solution requires the use of a node with a static public IP address, the two peers simply use the server to relay messages between them, Instead of attempting a direct connection.

The use of a relay, to implement a sort of hybrid peer-to-peer with client/server architecture is quite promising. The major advantages of using a relay in a P2P network are:

1) Clients can practically always connect to it, unlike the other two NAT-traversal techniques. It is considered the most reliable method of NAT-traversal because it ensures 100% connectivity between the peers (clients) as long as both peers have connectivity to the relay.
2) Relay does not require user intervention, unlike other strategies like port forwarding in which the users need to configure their home router.
3) Relay can allow much more control over the security of the solution, unlike port-forwarding.
4) Relay can massively reduce packet count and bandwidth consumption when multi-casting in peer-to-peer networks. Once the relay receives the message from the client, it sends it to each client. Another scenario where the use of a relay can avoid network congestion. In fact, when multiple users are all sending to the same user, in this case, the relay can combine their packets into a single packet and forward it to its destination.

On the other hand, relaying consumes the server's processing power and network bandwidth. Moreover, the use of a relay will increase the network latency, this is especially noticeable when the application is sensitive to delay like CVR. In order to benefit from the advantages of relay and adapt it to CVR applications, this article suggests an approach that leverages container migration to mitigate the drawbacks that come with relay, as a NAT-traversal solution, and accentuate its advantages.

IV. Our Solution

In an attempt to reduce the delay caused by relay servers while ensuring maximum reliability and failure management we took inspiration from the survey conducted by Qirtas and al [8]. The authors have studied the impact of several factors like the number of relay nodes, their placement, buffer management, the network topology on improving the performance of relays by minimizing the network latency. One of the most important factors discussed is the relay node's placement; nodes with fixed static placement have sub-optimal network performance, while mobile-based nodes that can change position according to the network topology and the connected participants' needs, can be more efficient because it increases the delivery ratio. Another issue that hybrid P2P applications (that are a combination of P2P and client-server models) struggle with it when the host (the peer that plays the role of both a client and a server) drops from the network causing all the other users to lose connection to the application. In this regard, we suggest a solution that allows the selection of the optimal host, and the migration of the host if the user in question drops from the network.

Our suggested approach contains the following models (see Figure 1):

A. Relay Service

Relay agents are intermediary elements that accept requests and transfer packets to other nodes based on information found in the messages. They also modify messages by inserting and removing routing information but do not modify any other portion of it. Relay servers may be used to aggregate requests from multiple peers, the use of relays also eliminates the need for peers to be configured with the necessary security information they would otherwise require to communicate with other peers behind a NAT/Firewall. The relay used here must be deployed in a container. In our case, we used Light Reflective Mirror [9], a free self-hosted relay available for all platforms (PC, Mac, Linux, WebGL, Android, IOS) and can be deployed using Docker in order to migrate the container dynamically, Fig 1 explains the general layout of our solution.

B. Node Migration Manager

Mechan and al. in [10] proved how promising container migration can be to improving the general performance in Mobile Edge computing, the authors presented a multilayer framework for migrating active applications to the nodes closer to clients. Container migration is the process of transferring a container from one source to a destination node. It can help provide fault tolerance in case of system failure, it serves also for load balancing, hardware breakdowns, scaling, and resource reallocation [11].

In order to ensure a smooth experience for the CVR users, we opted for live migration using the checkpoint/restore mechanism, Live migration is the mechanism of transferring a running container from one server to another with minimal interruption. The migration is performed by sending the state of the processor, network, and local storage in addition to memory and disk content to the destination host [12]. One live migration mechanism is checkpoint/restore [13], which is an efficient container-based service migration, that leverages the CRIU tool that allows to back up the state of a running container so that its execution can later be recovered from the time of the checkpoint. The **checkpointing** operation collects

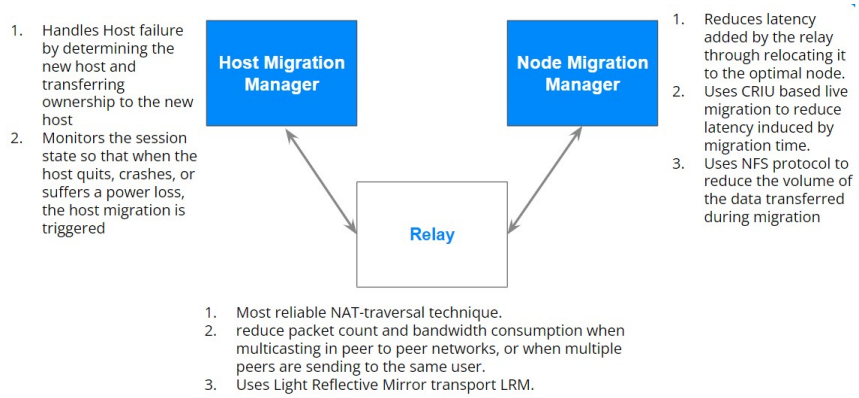


Fig. 1. General design of the suggested approach.

and retains the current status of all running processes in the container. Following that, all processes will be terminated and a new activity, restore, will be launched. The **restore** operation initiates the creation of new processes using the previously acquired dumped file at checkpoint, upon resuming the process execution and activating the network, the container resumes its usual operations. In order to avoid sending voluminous files that are the product of the checkpoint command across the network, we opted to use NFS storage to share and centralize the container file system.

In our context, the relay container consists of description image files, Runtime Memory, and log files, saving these files in the shared mount between the host and the destination instead of sending them through the network saves a lot of time and allows a faster restoration process. In a conventional setup, after the migration the container will be accessible through the new host node IP, this is disturbing for our already connected clients since the CVR application was built using the old host IP, in order to mend that and ensure the high availability of our architecture we will be using a **Floating IP** address shared between the Primary and secondary node, a floating IP is a virtual IP address that can be instantly moved from one node to another with passive/active configuration. When the restoration is completed and the relay container is running on the secondary node the Floating IP will be assigned to the node automatically and transparently, to avoid any additional downtime.

1) Choose optimal relay node location Algorithm(C.O.R.N.L): This algorithm is executed by the node migration manager every time a room is created in order to choose the optimal location for the relay container by comparing the average distance between the connected clients and the two available nodes. If the secondary node has an average distance inferior to the primary one, then the live migration is activated and the relay container will migrate to the secondary node. We used geophysical distances to approximate latency since there is a correlation between the two. The further node from another is, the higher latency between the two becomes, see Algorithm 2. In order to get clients' locations, we send an HTTP request to an IP geolocation website with web API for example http://ipinfo.io

Calculating the distance between two nodes: To determine the distance between the clients and nodes (algorithm 1) using their GPS coordinates (latitude and longitude), we will be using the **Haversine formula**

C. Host Migration Manager

For host migration to work properly, the session's currents state and context need to be saved, the data to save ranges from the users' progress to their character's statistics and position, to achieve this each client periodically is required to serialize the data in JSON format using the JSON.Net package, the unity web API is used to send the data to the host migration manager that stores all the session context (connected clients IPs, non-Player and player GameObjects,...etc.). Thus, when the new host is selected, the host migration manager sends the saved game context across the network, the new host deserializes the received JSON data to respawn the game objects and recreate the scene just like it was before the failure.

The session also needs to be monitored so the system can detect when the host loses connection, a way is to implement a host timeout detection mechanism by saving the timestamp of the last event received from the host and checking that value constantly by the Host migration process. When the manager detects that the Master peer is not sending events as expected it can then activate the Host migration process.
The steps can be enumerated into the following:

1) The timeout mechanism detects the host failure.
2) The Host migration manager sends a request to the participants to disconnect from the host and disables all GameObjects.
3) Determination of the new host process is activated and the new host is selected by H.M.M (Host Migration Manager), since we are transferring a considerable amount of data that describes the room's state, the new host is the client with the lowest latency.
4) The H.M.M sends the game saved state to the new host.
5) The new host deserializes the state and reconstructs the scene.
6) The rest of the players are instructed to reconnect to the new host and reactivate all GameObject.

Algorithm 1 Function cal_dist(node1,node2)

Data: node1(lat1,long1), node2(lat2,lon2) are the GPS coordinates (latitude,longitude) of two nodes we want to determine the distance between.
Result: the distance in meters between node1 and node2
$R \leftarrow 6371000$;
$DeltaPhi \leftarrow lat1 - lat2$;
$DeltaLambda \leftarrow long1 - long2$;
$a \leftarrow \sin^2(\frac{DeltaPhi}{2}) + \cos(lat1).\cos(lat2).\sin^2(\frac{DeltaLambda}{2})$
$c \leftarrow 2.\arctan^2(\sqrt{a}, \sqrt{1-a})$;
$dist \leftarrow R.c$
return dist

Algorithm 2 Algorithm C.O.R.N.E.L

Data: loc_prim_node, loc_secondary_node: coordinates of the primary node
and the secondary node.
Result: a Boolean that if set to true launches the live migration and else do nothing.
$sum\_dist\_prim\_node \leftarrow 0$
$sum\_dist\_second\_node \leftarrow 0$
$result \leftarrow False$
if room_started **then**
$connected\_clts \leftarrow get\_connected\_clts()$ /* this will send a request to
the relay container to retrieve the list of connected clients*/
$conncted\_clt \in connected\_clts$
$loc\_clt \leftarrow get\_location(connected\_clt.ip)$
$dist\_clt\_prim\_node \leftarrow cal\_dist(loc\_clt, loc\_prim\_node)$
$sum\_dist\_prim\_node \leftarrow sum\_dist\_prim\_node + dist\_clt\_prim\_node$
$dist\_clt\_second\_node \leftarrow cal\_dist(loc\_clt, loc\_secondary\_node)$
$sum\_dist\_second\_node \leftarrow sum\_dist\_second\_node + dist\_clt\_second\_node$

$avg\_dist\_prim\_node \leftarrow \frac{sum\_dist\_prim\_node}{connected\_clts.numberOfConnedtedClts}$
$avg\_dist\_second\_node \leftarrow \frac{sum\_dist\_second\_node}{connected\_clts.numberOfconnedtedClts}$
if $avg\_dist\_prim\_node > avg\_dist\_second\_node$ **then**
$result \leftarrow True$
end if
end if
return $result$;

To choose the new host among the connected clients, we use latency as a decision metric since we will be sending a considerable amount of data across the network. The new host should be the closest to the relay node. Refer to algorithm 3.

Algorithm 3 Algorithm Determination of New Host

Data: loc_node: coordinates of the relay node
Result: The IP address of the new host.
$dist \leftarrow 0$;
$min\_dist \leftarrow Max\_Value$; /*we initialise the min distance to the maximum value.
$new\_host \leftarrow 0.0.0.0$;
if host_Migration_activated **then**
$connected\_clts \leftarrow get\_connected\_clts()$
$conncted\_clt \in conected\_clts$
$loc\_clt \leftarrow get\_location(connected\_clt.ip)$
$dist \leftarrow cal\_dist(loc\_clt, loc\_node)$
if $dist <$ min_dist **then**
$min_dist \leftarrow$dist;
new_host ←conncted_clt.ipadress;

end if

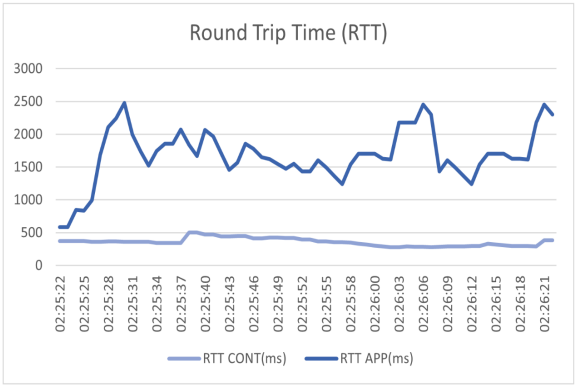
end if
return new_host;

V. Performance evaluation

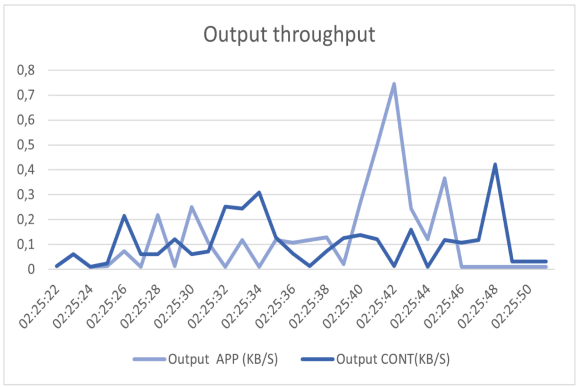
We have evaluated our framework by creating our experimental setup that consists of our different VMs. Two Linux VMs represent the primary node and secondary node, respectively. Whereas, the two widows VMS that represents our clients, $client_1$ and $client_2$, in a way that $client_1$ and $client_2$ are behind different LANs and geographically distant. To control the link speed and mimic the distance "tc" command has been employed. The experiment evaluation starts by launching the LRM-node in the primary node and connecting the two clients to it one as a host and the other as a client. We have used measuring tools to collect information about the performance, after collecting the data and treating them the next step is launching the migration operation, following the migration once more we collect performance measurements to identify the effect of our framework on the QoE. We have compared our solution to the baseline approach, whereby the relay server has a fixed location. In contrast, our solution deploys the relay server as a container that dynamically migrates from the primary node to the secondary one to ensure a better experience for connected participants.

A. Delay evaluation

We have compared our solution to the baseline approach in terms of delay (RTT: Round Trip Time). By definition, RTT is the time it takes for a packet to go from the sending endpoint to the receiving endpoint and back. It includes propagation delay, processing delay, queuing delay, and encoding delay. The higher the RTT value and the lower the service quality.



(a) Delay evolution of our solution compared to the baseline approach



(b) Throughput evolution of our solution compared to the baseline approach

Fig. 2. Performance evaluation of the proposed solution in terms of delay and throughput.

It is considered more accurate since it is measured at the application layer and includes the additional processing delay produced by higher-level protocols and applications.

Fig. 2(a) shows the performance evaluation of our solution (CONT) compared to the baseline approach (APP). The first observation that we can draw from this figure is that the RTT in the baseline fluctuates wildly raising from 584 ms to 2481 ms, while our RTT in our solution is relatively stable and lower than 400 ms. The obtained results demonstrate the efficiency of our solution in terms of end-to-end delay.

B. Throughout evaluation

Throughput is a network metric that indicates how much data an application produces to send across the network at any given time. Fig 2(b) shows the evaluation of our solution in terms of network throughput. From this figure, we observe that the throughput keeps fluctuating and varies from one-time interval to another. The main observation from this figure is that our solution generates lower bandwidth compared to the baseline approach, which demonstrates the efficiency of our solution in terms of bandwidth consumption strategy.

VI. Conclusions

In this paper, we have presented the requirements and characteristics of collaborative virtual reality systems including flexibility and multiple viewpoints, sharing context, ownership, awareness, communication and negotiation, consistency, and scalability. Followed by an in-depth inspection of the main network challenges that need to be addressed in order to fulfill the aforementioned requirements, and how failing to meet the conflicting and stringent constraints on bandwidth, latency and reliability can impact the user's experience of the collaborative virtual environment. We present the network architectures used in CVR systems, their limitations, and how relay-based network architectures can contribute to accomplishing the ideal vision for CVR systems. We have proposed a new solution that leverages the network relay service and container migration for enhancing the quality of experience (QoE) by reducing the delay and enhancing bandwidth utilization. The obtained results demonstrate the efficiency of the proposed solution in terms of end-to-end delay and bandwidth.

VII. Acknowledgements

This work has been partially supported by the European Union's Horizon 2020 Research and Innovation program, under the project ACCORDION (Grant agreement ID: 871793).

References

[1] "A blueprint of technology, applications and market drivers towards the year 2030 and beyond," *ITU-T FG-NET-2030, ITU*, Geneva, Switzerland, May 2019.

[2] M. R. Macedonia and M. J. Zyda, "A taxonomy for networked virtual environments," *IEEE multimedia*, vol. 4, no. 1, pp. 48–56, 1997.

[3] M. S. Elbamby, C. Perfecto, M. Bennis, and K. Doppler, "Toward low-latency and ultra-reliable virtual reality," *IEEE Network*, vol. 32, no. 2, pp. 78–84, 2018.

[4] E. Bastuğ, M. Bennis, M. Médard, and M. Debbah, "Towards interconnected virtual reality: Opportunities, challenges and enablers," *arXiv e-prints*, pp. arXiv–1611, 2016.

[5] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, "Edge-centric computing: Vision and challenges," pp. 37–42, 2015.

[6] H. Bandara and A. P. Jayasumana, "Collaborative applications over peer-to-peer systems–challenges and solutions," *Peer-to-Peer Networking and Applications*, vol. 6, no. 3, pp. 257–276, 2013.

[7] S. Bryson, "Approaches to the successful design and implementation of vr applications," *Virtual reality applications*, pp. 3–15, 1995.

[8] M. M. Qirtas, Y. Faheem, and M. H. Rehmani, "Throwboxes in delay tolerant networks: A survey of placement strategies, buffering capacity, and mobility models," *Journal of Network and Computer Applications*, vol. 91, pp. 89–103, 2017.

[9] "Derek-r-s/dark-reflective-mirror: A relay transport for mirror using darkrift2 as the relay server." https://github.com/Derek-R-S/Dark-Reflective-Mirror, (Accessed on 02/08/2022).

[10] A. Machen, S. Wang, K. K. Leung, B. J. Ko, and T. Salonidis, "Live service migration in mobile edge clouds," *IEEE Wireless Communications*, vol. 25, no. 1, pp. 140–147, 2017.

[11] P. Niroj, "Live container migration: Opportunities and challenges," *Aalto University*, 2017.

[12] S. V. N. Kotikalapudi, "Comparing live migration between linux containers and kernel virtual machine: investigation study in terms of parameters," 2017.

[13] R. A. Addad, D. L. C. Dutra, M. Bagaa, T. Taleb, and H. Flinck, "Towards a fast service migration in 5g," in *2018 IEEE Conference on Standards for Communications and Networking (CSCN)*. IEEE, 2018, pp. 1–6.