§Grant Agreement No.: 101016509
Research and Innovation action
Call Topic: ICT-40-2020: Cloud Computing

CHARITY
Cloud for Holography and Augmented RealITY

Cloud for Holography and Cross Reality

D4.4: Evaluation, Validation and Showcasing Outcomes (preliminary)

Version: v1.0

Deliverable type	R (Document, report)
Dissemination level	PU (Public)
Due date	31/01/2024 (Latest Amendment)
Submission date	29/02/2024
Lead editor	Maria Pateraki (ORAMA)
Authors	Antonis Protopsaltis (ORAMA), Zbigniew Ledwoń (ORBK), Enrico Zschau (SRT), Luís Rosa (ONE), Luís Cordeiro (ONE), Theodoros Theodoropoulos (HUA), Antonios Makris (HUA), Konstantinos Tserpes (HUA), Massimo Coppola (CNR), Alex Roibu (HOLO3D), Laura Sande (PLEXUS), Joao Rodriguez (DOTES), Mike McElligott (UTRC), Peter Gray (CS), Ferran Diego Antilla (TID), Tarik Taleb (ICT-FI), Tarik Benmerar (ICT-FI)
Reviewers	Massimiliano Corsini (CNR), Diogo Fevereiro (ONE)
Work package, Task	WP4, T4.3
Keywords	Evaluation, use cases

Abstract

This document reports on the outcomes of the evaluation, validation and showcasing activities. It reports on the first set of experiments, technological setups and validation, to provide feedback to technological WPs and on the impact the evaluation and validation had on the development process. A combination of different subtopics with related metrics is exploited, derived primarily from log data in order to assess individual functionalities of the platform components/services as well as Use Case related aspects and verify the functional aspects of their intended operation. This document constitutes the first of two versions of the validation and evaluation.

Document revision history

Version	Date	Description of change	List of contributor(s)
v0.1	09/10/23	Initial ToC	ORAMA
v0.2	19/11/23	Evaluation subtopics definition	ORAMA
v0.3	27/12/23	Section 3 inputs	CNR, DOTES, HUA, ONE, ORBK, ORAMA, PLEXUS, UTRC
v0.4	22/01/24	Updates by all partners	CNR, DOTES, HUA, HOLO3D, ONE, ORAMA, ORBK, PLEXUS, SRT, UTRC
V0.5	07/02/24	Updates by all partners	CNR, DOTES, HUA, HOLO3D, ONE, ORAMA, ORBK, PLEXUS, SRT, UTRC
v0.6	23/02/24	Updates	CNR, ORAMA, PLEXUS
v0.9		Reserved for version sent for GA approval	
v1.0	29/02/24	Reserved for the final version submitted to EC	

Disclaimer

Acknowledgment

The research conducted by CHARITY receives funding from the European Commission H2020 programme under Grant Agreement No 101016509. The European Commission has no responsibility for the content of this document.

[1] http://creativecommons.org/licenses/by-nc-nd/3.0/deed.en_US

Executive Summary

The deliverable delves into the comprehensive process of validating and evaluating the CHARITY prototype, a crucial step in ensuring its functionality and efficacy. It underscores the significance of assessing not only the individual components and services but also their integration within the broader context of the project's use cases. This evaluation takes place across diverse testbed environments provided by partners, enabling controlled tests and data collection essential for thorough analysis.

To gauge the success of the project's ambitions and methodologies, a multifaceted approach is employed, leveraging various metrics primarily derived from log data. These metrics serve as quantitative indicators, shedding light on the technical characteristics and operational functionalities of the CHARITY platform. By scrutinizing these metrics, valuable insights into the platform's performance and its alignment with project objectives is gained.

The document delineates the validation and evaluation process into distinct phases, with this particular report representing the initial stage. It focuses on reporting the outcomes of the first set of experiments, technological setups, and validation procedures. Moreover, it provides valuable feedback to the technological work packages, informing future development endeavors.

A critical aspect of the evaluation involves defining and addressing specific subtopics linked to project requirements, both functional and non-functional. Each subtopic is meticulously examined, with appropriate metrics defined to measure its performance. Experimental procedures are outlined, detailing the methodology, tools, and instruments utilized to gather relevant data during functional tests.

Furthermore, the document emphasizes the necessity of defining testbed characteristics and capacities to facilitate the deployment of CHARITY components. This involves compiling detailed descriptions of the testbed infrastructure, including production cloud resources and open-source cloud stacks. Such information is crucial for ensuring seamless integration and optimal performance across diverse environments.

Overall, the deliverable serves as a comprehensive guide to the validation and evaluation process, providing invaluable insights into the progress and performance of the CHARITY prototype.

Table of Contents

Executive Summary **3**
Table of Contents **4**
List of Figures **6**
List of Tables **7**
Abbreviations **8**
1 Introduction **10**
1.1 Scope, Motivation and Objectives 10
1.2 Methodology 10
1.3 Structure of the document 10
2 Preparatory activities for evaluation **11**
2.1 Evaluation subtopics definition 11
2.1.1 Platform 11
2.1.2 XR services 11
2.1.3 Use Cases 11
2.2 Procedures and metrics definition 12
2.3 Testbeds and resources 12
2.3.1 CloudSigma Testbed Characteristics and Capacity 13
2.3.2 TID Testbed Characteristics and Capacity 15
2.3.3 OneSource Testbed Characteristics and Capacity 16
3 Evaluation and results **18**
3.1 High level orchestrator (CNR) 18
3.1.1 Description, procedure, metrics 18
3.1.2 Tests, data collection and analysis 19
3.2 Low level orchestrator (ONE, ICT-FI) 20
3.2.1 Description, procedure, metrics 20
3.2.2 Tests, data collection and analysis 20
3.3 Monitoring Framework (PLEX) 21
3.3.1 Description, procedure, metrics 21
3.3.2 Tests, data collection and analysis 22
3.4 Forecasting Model (HUA) 23
3.4.1 Description, procedure, metrics 23
3.4.2 Tests, data collection and analysis 23
3.5 Point Cloud Encoding/Decoding (CNR) 24
3.5.1 Description, procedure, metrics 24

3.5.2 Tests, data collection and analysis 25
3.5.3 KPIs assessment 25
3.6 Mesh Merger (CNR) 26
3.6.1 Description, procedure, metrics 26
3.6.2 Tests, data collection and analysis 26
3.6.3 KPIs assessment 27
3.7 UC1-1 Holographic Concert and UC1-2 Holographic meetings (HOLO3D) 27
3.7.1 Description, procedure, metrics 27
3.7.2 Tests, data collection and analysis 28
3.7.3 KPIs assessment 30
3.8 UC1-3 Holographic Assistant (SRT) 30
3.8.1 Description, procedure, metrics 30
3.8.2 Tests, data collection and analysis 31
3.8.3 KPIs assessment 32
3.9 UC2-1 VR Medical Training (ORAMA) 32
3.9.1 Description, procedure, metrics 32
3.9.2 Tests, data collection and analysis 34
3.9.3 KPIs assessment 35
3.10 UC2-2 VR Tour Creator (DOTES) 36
3.10.1 Description, procedure, metrics 36
3.10.2 Tests, data collection and analysis 37
3.10.3 KPIs assessment 39
3.11 UC3-1 Collaborative Gaming (ORBK) 40
3.11.1 Description, procedure, metrics 40
3.11.2 Tests, data collection and analysis 41
3.11.3 KPIs assessment 42
3.12 UC3-2 Manned-Unmanned Operation Trainer (UTRC) 42
3.12.1 Description, procedure, metrics 42
3.12.2 Tests, data collection and analysis 43
3.12.3 KPIs assessment and requirement satisfaction 50
4 Lessons learnt 52
5 Conclusions 55

List of Figures

Figure 1 - Orchestrator Metrics Dashboard showcased at EUCnC 2023 21

Figure 2. A test scene reconstructed from 8 RGBD views. 25

Figure 3. Network load with 1280x720 resolution @25fps and an average 2500kbps 29

Figure 4. Latency with 1280x720 resolution @25fps and an average 2500kbps 30

Figure 5. Metrics from Test 1. The latency in ms (upper left), the frames per second (upper right), the packet lost % (lower left) and the sent rate (lower right). 34

Figure 6. Metrics from Test 2. The latency in ms (upper left), the frames per second (upper right), the packet lost % (lower left) and the sent rate (lower right). 35

Figure 7. Metrics from Test 3. The latency in ms (upper left), the frames per second (upper right), the packet lost % (lower left) and the sent rate (lower right). 35

Figure 8. Bytes received 37

Figure 9. Bytes sent 38

Figure 10. Total Round Trip 38

*Figure 11.*Current Round Trip 38

Figure 12. Available incoming bitrate 39

Figure 13. Measured network latency data: Game Clients <-> Game Servers and Game Servers <-> Mesh Mergers 41

Figure 14. Measured computational latency data (RTT): Game Clients <-> Game Servers and Game Servers <-> Mesh Mergers 42

Figure 15.Predicted trajectory versus observed trajectory 46

Figure 16. Diversity of configuration channels for remote rendering components 46

Figure 17. Dynamic Software Adaptation using rolling updates for the Collins use case 47

Figure 18. Generate high quality at rendering source 49

Figure 19. Generate low quality on the cloud and seek to recover quality at the edge. Significant bandwidth reductions but also significantly increased resource usage overall 49

List of Tables

Table 1. Summary of subtopics 12
Table 2. Testbed Template 13
Table 3. CloudSigma Testbed Characteristics 14
Table 4. TID Testbed Characteristics 15
Table 5. OneSource Testbed Characteristics 16
Table 6. Description of evaluation subtopic - High level Orchestrator 18
Table 7. Status and purposes of metrics data gathered in HLO testing 19
Table 8. Description of evaluation subtopic - Low level Orchestrator 20
Table 9. Description of evaluation subtopic - Monitoring Framework 22
Table 10. Description of evaluation subtopic - Forecasting model 23
Table 11. Experimental comparison between the proposed proactive horizontal scaling approach and the standard reactive one. Timings are in seconds. 24
Table 12. Experimental comparison in terms of Fault Tolerance 24
Table 13. Description of evaluation subtopic - Point Cloud Encoding/Decoding service 24
Table 14. Description of evaluation subtopic - Mesh Merger service 26
Table 15. Description of evaluation subtopic - Holographic Concert and Holographic meetings 27
Table 16. Description of evaluation subtopic - Assistant providing holographic 3D visual and spoken information 30
Table 17. Description of evaluation subtopic - Realistic simulation in VR medical training 32
Table 18. Description of evaluation subtopic - Virtual Experiences Builder for the web 36
Table 19. Description of evaluation subtopic - Mobile multiplayer game utilising AR technology 40
Table 20. Description of evaluation subtopic - Cloud Native Flight Simulator 42
Table 21. Experimental results of evaluating upscaling techniques 44
Table 22. Resource usage profiles across various configurations 47
Table 23. Requirements status for Flight Simulator Use Case 50

Abbreviations

5G-PPP	5G Infrastructure Public Private Partnership
AAA	Authentication, Authorization, Accounting
AAE	Adversarial Autoencoder
AI	Artificial Intelligence
API	Application Programming Interface
AR	Augmented Reality
CCU	Concurrent User
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
DASH	Dynamic Adaptive Streaming over HTTP
EDSR	Enhanced Deep Super Resolution
ESPCN	Efficient Sub-Pixel Convolutional Neural Network
ESRGAN	Enhanced Super-Resolution Generative Adversarial Network
ETSI	European Telecommunication Standard Institute
FSRCNN	Fast Super-Resolution Convolutional Neural Network
GPU	Graphics Processing Unit
GSM	Game Server Manager
HD	High Definition
HDD	Hard Disk Drive
HLO	High Level Orchestrator
HLS	Http Live Streaming
HMD	Head-Mounted Display
HPC	High Performance Computing
IPFT	Intelligent proactive Fault Tolerance
JSON	Javascript Object Notation
LAPSRN	Laplacian Pyramid Super-Resolution
LHLS	Low Latency Http Live Streaming
LLO	Low Level Orchestrator
LSTM	Long Short-Term Memory
MILP	Mixed-Integer Linear Programming
PLY	Polygon File Format
RIFE	Real-Time Intermediate Flow Estimation
QoE	Quality of Experience
QoS	Quality of Service
RAM	Read Access Memory
REST-API	Representational State Transfer Applicaiton Programming Interface
RFT	Reactive Fault Tolerance
RTMP	Real-Time Messaging Protocol

RTSP	Real-Time Streaming Protocol
RTT	Round Trip Time
SRGAN	Super Resolution Generative Adversarial Network
SSD	Solid State Drive
UC	Use Case
UDP	User Datagram Protocol
VDI	Virtual Desktop Infrastructure
VPN	Virtual Private Network
vGPU	Virtual Graphics Processing Unit
VM	Virtual Machine
VR	Virtual Reality
XR	Extended Reality
ZSM	Zero Touch Management

1 Introduction

1.1 Scope, Motivation and Objectives

The CHARITY approach is based on a two-stage prototyping and evaluation cycle which focuses on researching, designing, implementing and evaluating a cloud-native framework to be able to use specific mechanisms to support the deployment and life-cycle management of a set of Cloud-based XR Services (e.g., distributed holographic, AR and VR applications) and improving the overall applications' performance and user experience.

The deliverable focuses on the validation and evaluation of components and services of the CHARITY prototype as well as the Use Cases in different testbed environments. The testbed environments are provided by the partners and allow the execution of controlled tests and collecting the required measurements for the assessment. The validation and evaluation aim to provide insight to whether the project ambitions and approaches provide tangible outcomes to the use cases of the projects. A combination of metrics is exploited, derived primarily from log data to derive conclusions on the platform's technical characteristics and functionalities.

This document constitutes the first of two versions of the validation and evaluation, reporting on the first set of experiments, technological setups and validation, aiming to provide feedback to technological WPs and on the impact the evaluation and validation had on the development process.

1.2 Methodology

We mainly devised the content presented in this deliverable based on the following approaches:

- Via regular communication between CHARITY partners using suitable communication tools (e.g., cross Work Packages meetings, offline communication through e-mail)
- Sharing examples to guide partners on drafting their evaluation subtopics and providing different rounds of feedback to support a common approach and alignment between the different subtopics handled
- Each partner responsible for an evaluation subtopic has conducted the required functional tests and experiments to collect and analyse relevant data. For the evaluation subtopics related to the use cases reference to relevant KPIs was provided.

1.3 Structure of the document

Section 2 reports on the preparatory activities for evaluation with reference to the evaluation subtopic definition. The subtopics were linked to the requirements devised in D1.2 and categorized according to platform components, services and use cases. The section further reports on the testbeds and resources that will be used for experimentation and validation.

Section 3 details the individual subtopics, the tests, the data collection and analysis considered in this phase.

Section 4 summarizes the lessons' learnt.

2 Preparatory activities for evaluation

2.1 Evaluation subtopics definition

In order to assess individual functionalities of the platform components/services as well as Use Case related aspects and verify the functional aspects of their intended operation a number of related subtopics was provided. The subtopics were linked to the requirements devised in D1.2 and the non-functional requirements addressing different attributes of the system. For each subtopic appropriate metrics were defined and the necessary tools have been utilized by each partner to gather the relevant evaluation data during the functional tests (i.e., log data).

2.1.1 Platform

As reported in D4.2 the integrated components will be evaluated as part of T4.3 and the efforts of this task will be reflected in two document versions, namely D4.4 and D4.6 focused on reporting all the experimentation testing outcomes. As part of this deliverable the following platform components related to separate evaluation subtopics were evaluated:

- The High-level orchestrator – responsible for deciding where to allocate resources to accommodate XR services or when and where to migrate a component to a new location.
- The Low-level orchestrator - responsible for provisioning and orchestrating a multi-domain infrastructure and XR services within each cluster.
- The monitoring framework inspired by the underlying idea of Zero touch network & Service Management (ZSM) from ETSI and designed as a dynamic architecture capable of coping with infrastructure and network changes.
- The forecasting model as part of the Resource Usage Prediction Mechanism and the Service Traffic Prediction Mechanism.

2.1.2 XR services

As reported in D3.2 specific data services were developed as part of the project and are exploited by a subset of XR applications of the CHARITY project. Even if these data services are targeted to the use cases of CHARITY, it is envisioned by the partners of the consortium that such services can be used/adopted by other XR applications with similar needs, beyond the ones involved in the project itself. As part of this deliverable the following XR services were evaluated:

- The Mesh Merger service which employs geometry processing algorithms to build virtual environment for AR applications, that enables the UC3-1 Collaborative Game
- The Point cloud encoder/decoder service, that is the main component of the UC1-3 Holo Assistant and supports the efficient transmission of a huge amount of 3D data from the cloud to the edge (the holographic display).

2.1.3 Use Cases

All the application Use Cases and developed components following the integration and experimentation plan reported in Section 5 of D4.2 have been evaluated in this deliverable. With the exception of the Use Case UC1-3 Holographic Assistant (due to SRT leaving the consortium) all other Use cases will also be evaluated in the upcoming final version of the deliverable, exploiting the deployment, monitoring and orchestration features of the CHARITY platform.

2.2 Procedures and metrics definition

For each of the listed subtopics, proper experimental procedures and metrics are defined. The main items related to the evaluation subtopics definition were focused on providing information on a) the evaluation scope of each subtopic, b) the related requirements, c) relevant platform application components, d) the measurement points for collecting and storing the data, e) the instruments and tools exploited, f) the methodology and procedure and g) the metrics to analyse the results. *Table 1* provides a summary of the subtopics that are handled in this deliverable, along with a reference to which category they belong (platform, XR services, Use Cases) and the metric used in the evaluation. Detailed descriptions and results of the individual subtopics are reported in Section 3.

Table 1. Summary of subtopics

Subtopic description	**Category**	**Metric/Evaluation**
High level orchestrator	Platform	Different metrics
Low level orchestrator	Platform	Different metrics
Monitoring Framework	Platform	Different metrics
Forecasting model	Platform	Different metrics
Point Cloud encoding/decoding	XR services	Different metrics, KPIs
Mesh Merger	XR services	Different metrics, KPIs
Holographic Concert and holographic meetings	Use case	Different metrics, KPIs
Assistant providing holographic 3D visual and spoken information	Use Case	Different metrics, KPIs
Realistic simulation in VR medical training	Use Case	Different metrics, KPIs
Virtual Experiences Builder for the web	Use Case	Different metrics, KPIs
Mobile multiplayer game utilising AR technology	Use Case	Different metrics, KPIs
Cloud Native Flight Simulator	Use Case	Different metrics, KPIs

2.3 Testbeds and resources

This section describes the general testbed characteristics and capacity of the combined testbed infrastructure consisting of CloudSigma's production cloud and supplementary testbed deployments. We detail each operator's general characteristics, resource capacity, account creation and access criteria. We also describe the ongoing technical support and maintenance required to ensure continuous operation throughout the project.

The main objective is to support the integration of the components developed in WP2 and WP3, according to the technical requirements of the CHARITY architecture and provide the underlying infrastructure to deliver a working proof-of-concept for validation and demonstration.

To support the deployment of CHARITY components on the combined testbed infrastructure, we must first define testbed descriptions with the characteristics and capacity available per provider/operator. This information is collected using *Table 2* as a template, which will be kept up to date throughout the project's duration. As stated, the CHARITY testbed combines production cloud resources (e.g., CloudSigma) with private clouds based on widely used open-source cloud stacks (e.g., Openstack).

Table 2. Testbed Template

Short description	
General configuration	
Hypervisor	
IaaS stack/version	
VM Monitoring	
Access methods	
Connectivity	
Cloud interface	
Provisioning	
Integration/drivers	
Networking	
Compute capacity (available for project use)	
CPU (Ghz)	
RAM (GB)	
Number of VMs	
Storage capacity (available for project use)	
SSD (GB)	
HDD (GB)	
Image format	
Networking	
Max internal network bandwidth per VM (Gb)	
Max external network bandwidth per VM (Gb)	
Max inter-VM latency (ms)	
Total cloud external network bandwidth (Gb)	

2.3.1 CloudSigma Testbed Characteristics and Capacity

Leveraging the expertise of CloudSigma, we have already identified one initial testbed deployment at CloudSigma premises (c.f. *Table 3*). This testbed is a critical milestone that helped the overall consortium understand the appropriate technologies, requirements, and directions to consider. The locations and characteristics of the remaining testbeds will result from an ongoing discussion regarding the exact needs of each task, the CHARITY concept, and the components to be evaluated.

CloudSigma has been working on GPU integration at its several European cloud locations. At the time of writing, we have already installed NVIDIA RTX A6000 graphics cards in CloudSigma Swedish cloud location. Installing NVIDIA A100 Tensor Core GPUs is also possible if the specifications are better suited to the use case requirements. CloudSigma made available a few GPU-equipped VMs to partners in Q3 2023.

The NVIDIA RTX A6000 GPU is ideal for data analytics workloads and applications like VDI, high-performance computing (HPC), and AI/Deep learning. The NVIDIA A100 Tensor Core GPU is optimized for HPC, AI, and data analytics. However, it is essential to note that some advanced settings will not

yet be available via the CloudSigma Web Interface with both GPU models. CloudSigma will expose these features in the coming months.

CloudSigma has also explored two possibilities for allocating GPU resources and sharing among virtual machines or containers in a virtualised environment: passthrough and vGPU (virtualised GPU). Passthrough is typically preferred for workloads that require maximum GPU performance, such as high-performance computing or deep learning applications, but require dedicated access to the GPU. At the same time, vGPU is more suitable for scenarios where GPU resources need to be shared among multiple VMs or containers, such as VDI or multi-tenant environments, offering a balance between performance and resource consolidation. While CloudSigma has successfully tested both options, only passthrough is enabled at the time of writing, meaning project partners can only attach one GPU per VM. CloudSigma plans to increase this to 2, 4, 8 and 16 in the coming months. vGPU capabilities will also be rolled out in the coming months.

Table 3. CloudSigma Testbed Characteristics

CloudSigma Cloud Locations: Geneva, Switzerland (GVA) and Boden, Sweden (LLA)	
Short description	Production IaaS platforms in Geneva (GVA) and Boden (GVA). The platform combines a proprietary stack with open-source technologies to provide a utility approach to IaaS provisioning. The platform offers a high level of control and flexibility in the provision of computational power, RAM, storage, and networking.
General configuration	
Hypervisor	KVM
IaaS stack/version	Proprietary CloudSigma stack
VM Monitoring	Intra-VM testing tools, at the discretion of the VM owner, NewRelic third-party integration
Access methods	API via HTTPS
Connectivity	Internet, VPN, Secure Remote User Access, Direct private patch to local switch
Cloud interface	WebApp, API
Provisioning	API, API middleware, WebApp, Python library (Pycloudsigma).
Integration/drivers	Ansible, CloudInit, Apache Libcloud, JClouds, Fog, Abiquo Hybrid Cloud, pycloudsigma Library, Terraform, Cluster API
Networking	API, WebApp
Compute capacity (available for project use)	
CPU (Ghz)	100Ghz
RAM (GB)	100GB
vGPU (instance spec.)	Mtultiple (20 x NVIDIA A6000 min.)
Number of VMs	Unlimited
Storage capacity (available for project use)	
NVMe SSD (GB)	3000
HDD (GB)	N/A
Image format	RAW
Networking	
Max internal network bandwidth	20

per VM (Gb)	
Max external network bandwidth per VM (Gb)	10
Max inter-VM latency (ms)	1

2.3.2 TID Testbed Characteristics and Capacity

Leveraging TID expertise, a second testbed deployment was already identified at TID premises (Valladolid) (c.f. *Table 4*). This testbed is also an important milestone that helps test the developed technologies with huge computational resource requirements in a customized infrastructure with GPU support.

TID has been working on installing a less powerful workstation in TID offices (Barcelona) to test onsite components with extremely low latency requirements. At the time of writing, TID has installed NVIDIA RTX 3090 graphics cards in both locations. TID is increasing the number of VMs per cluster. Currently, one VM is supported. NVIDIA RTX 3090 GPU is ideal for data analytics workloads and applications like VDI, HPC, and AI/Deep learning. Being a testing platform allows one to test different GPU settings easily.

Table 4. TID Testbed Characteristics

TID Cluster Location (Valladolid)	
Short description	Test IaaS workstation in Valladolid (Zrh). The workstation uses open-source technologies to provide a utility approach to IaaS provisioning. The workstation offers a large computational resource with GPU support.
General configuration	
Hypervisor	-
IaaS stack/version	OpenStack
VM Monitoring	-
Access methods	ssh
Connectivity	Internet, VPN, Secure Remote User Access
Cloud interface	-
Provisioning	API, Python library
Integration/drivers	Flexible
Networking	N/A
Compute capacity (available for project use)	
CPU (Ghz)	100Ghz
RAM (GB)	126GB
vGPU (instance spec.)	3x Titan RTX 3900
Number of VMs	Currently up to 1 VM
Storage capacity (available for project use)	
SSD (GB)	500
HDD (GB)	9Tb
Image format	RAW
Networking	

Max internal network bandwidth per VM (Gb)	N/A
Max external network bandwidth per VM (Gb)	N/A
Max inter-VM latency (ms)	N/A

2.3.3 OneSource Testbed Characteristics and Capacity

Leveraging the expertise of OneSource, a third testbed has already been identified at OneSource premises to support further the integration and experimentation of the CHARITY framework. This testbed features a robust 3-node Kubernetes cluster (refer to *Table 5*), vital in facilitating the testing and evaluation of distributed scenarios encompassing hybrid edge-cloud domains. Additionally, it enables the exploration of multi-cluster deployments utilizing cutting-edge overlay networking technologies, such as Liqo. Furthermore, it offers an opportunity to assess further the performance and effectiveness of the low-level orchestrator within the CHARITY framework.

Table 5. OneSource Testbed Characteristics

OneSource Coimbra, Portugal	
Short description	Testbed is located in OneSource's datacenter. The testbed comprises a 3-node Kubernetes cluster for CHARITY project experimentation and validation.
General configuration	
Hypervisor	VMWare ESXi
IaaS stack/version	N/A
VM Monitoring	N/A
Access methods	Kubectl
Connectivity	VPN
Cloud interface	-
Provisioning	N/A
Integration/drivers	N/A
Networking	N/A
Compute capacity (available for project use)	
CPU (Ghz)	100Ghz
RAM (GB)	16
vGPU (instance spec.)	0
Number of VMs	3
Storage capacity (available for project use)	
SSD (GB)	50
HDD (GB)	N/A
Image format	-
Networking	
Max internal network bandwidth per VM (Gb)	-
Max external network bandwidth	-

per VM (Gb)	
Max inter-VM latency (ms)	-

3 Evaluation and results

3.1 High level orchestrator (CNR)

3.1.1 Description, procedure, metrics

Table 6. Description of evaluation subtopic - High level Orchestrator

Subtopic Title: High level Orchestrator	**Partners:** CNR
Short description and evaluation scope: The High-Level Orchestrator (HLO) acquires information about the application (via the UI) and the platform status (via the Monitoring and Forecasting modules) in order devise optimal plans for application allocation and enact them leveraging the Low-level orchestrator. Two scenarios are typically distinguished, (a) one where the application is initially deployed, and (b) a more complex one where deployment plan adaptation is triggered on a running application. In both cases the known available resources may either be enough to devise a deploy plan (sub-scenarios a.1, b.1) or interaction(s) with the platform may be needed to recruit additional resources (a.2, b.2). The evaluation of the HLO addresses these 4 main cases according to the following profiles i. Functional evaluation of the integration: interfacing with the surrounding components need to support all the sub-scenarios ii. Functional evaluation of the HLO "Solver" strategy: assessing the quality of the deployment solution(s) iii. Non-functional evaluation: assessing the performance of the HLO in terms of its Service Time (time to deploy and application) and throughput (sustained number of deploy requests per unit of time).	
Components involved: Monitoring Manager, Low Level Orchestrator	
Where are data collected and stored – measurement points: The tests are designed to work on both the actual platform deployed on Cloud Sigma premises, and in a simulated environment where the surrounding components are stubs allowing to perform intensive tests on the HLO alone, allowing faster testing and higher test throughput with a lower footprint in resources. The measurement points are mainly on the interfaces of the HLO with the surrounding components. Evaluating the Solver performance (i.e., both functional testing ii. and non-functional testing iii.) is entwined with the research on deploy optimization and requires through and careful analysis of the data, which will be the topic of scientific papers.	
When are data collected? The data collection will be performed in the 1st quarter of 2024. The test groups i. and iii. can be completed in a few days, while fully testing for group ii. may require further effort via simulated execution.	
Instruments/tools: The metrics will be gathered in two ways: 1. via direct instrumentation of the HLO code, allowing to measure execution times, delays, and overheads as well as to record the information streams to and from the HLO component, 2. by analysing the recorded input requests, input information and output plans, performed offline, in order to evaluate the qualitative aspects of the deployment.	
Methodology/Procedure: A series of tests will be performed on the platform in a default state, which request a fixed set of applications, with pseudo-randomized key request parameters and application execution order, to allow for significant and repeatable experiments. The execution order will cause different load condition on the platform and trigger the HLO behaviours.	
Metrics to analyze the results Service time (time to resent a deploy plan to the platform) in the different cases (e.g., initial deploy, adaptive redeployment, with/without request for additional resources). Deploy cost in terms of resources and resource cost. Effectiveness in matching the application stipulated QoS and constraints in input (e.g. as number and duration of QoS	

violations).

3.1.2 Tests, data collection and analysis

Preliminary testing of the HLO algorithms and solutions as integrated with the platform is still in progress. While the effectiveness of the solution has been already demonstrated with simulation and with simple platforms, more data is being gathered to analyse the support with an increasing number of virtual clusters (more complex platform) as well as number and range of deployed applications (including both blueprints of example applications and those from the project use cases). The following *Table 7* outlines the status of testing as of M37 of the project.

Table 7. Status and purposes of metrics data gathered in HLO testing

	A - Initial application deploy		B - Application redeploy at run time [1]	
	A.1 available resources are sufficient	A.2 available resources are not sufficient	B.1 available resources are sufficient	B.2 available resources are not sufficient
i. Functional evaluation (integration)	verified Related interfaces have been tested	verified Related interfaces have been tested	verified Related interfaces have been tested	verified Related interfaces have been tested
ii. Functional evaluation (Solver)	verified Optimal solution is generated	verified Detected condition triggers a request for more virtual clusters	preliminary Optimal solution is generated; eval. constraints about solution stability (reoptimization) needs more data from a larger platform	preliminary Detected condition triggers a request for additional virtual clusters; choosing ideal request for additional resources requires a meta-strategy, eval. data still being gathered
iii. Non-functional evaluation [2]	preliminary Now gathering a larger dataset [3]	preliminary Now gathering a larger dataset [3]	preliminary Now gathering a larger dataset [3]	preliminary Now gathering a larger dataset [3]

[1] due to any alert type, i.e., both platform alerts and application-specific ones

[2] including performance and scalability of both the platform integration and the Solver algorithms

[3] a larger dataset is to be prepared (a) collecting results for a multi-cluster environment where several applications are deployed, (b) including metrics for API call and dispatch overhead, service time of the HLO shell and of the MILP Solver module, amount of missing resources / in-excess allocated resources, service time for alerts and triggered redeploy, service queue length for *deploy* and *redeploy* actions.

The preliminary data collected is being organized as a dataset for use within the consortium as well as for future research on orchestration methods. Beside being used to support research papers by the partners, we will evaluate the options for curating it into an open dataset collection under the FAIR principles.

3.2 Low level orchestrator (ONE, ICT-FI)

3.2.1 Description, procedure, metrics

Table 8. Description of evaluation subtopic - Low level Orchestrator

Subtopic Title: Low Level Orchestrator	**Partners:** ONE/ICT-FI
Short description and evaluation scope: The Low Level Orchestrator is evaluated in two different scopes, *functional and non-functional (e.g., performance)*. In the functional scope, we evaluate the Low Level Orchestrator capabilities focused on their functional behaviour and ability to perform its actions/tasks (e.g., the ability to create a cluster and the deployment of applications). In the performance scope, we perform a quantitative evaluation on the time and resources used to achieve the aforementioned capabilities(e.g., how much time/CPU/RAM it takes to create a cluster).	
Components involved: Low Level Orchestrator	
Where are data collected and stored – measurement points: These tests were designed for CloudSigma testbed composed of two parts. First, a Kubeadm cluster for the deployment of the Low Level Orchestrator components (i.e., Backend, Operator, CRD). Second, a Openstack deployment for hosting the VMs and clusters to be created by the *LLO*. For the Openstack, we use a *Microstack setup*. Therefore, the measurement and evaluation will focus on these two points.	
When are data collected? The data will be collected during the 1st quarter of 2024.	
Instruments/tools: *which modules or questionnaire* Python3 for instrumentation of reproducible scripts; Prometheus, Kube-state-metrics and Kubernetes Metrics Server for applications and cluster-related metrics (e.g., resource consumption).	
Methodology/Procedure: *in which way data are collected* Each test consists of 10 runs, of which we aggregate the results. The time each run takes to complete may vary and as such, will be taken into account during the measurements and result analysis.	
Metrics to analyze the results Average Time/CPU Usage/RAM Usage to Create/Scale/Delete a cluster, Average Time/CPU Usage/RAM Usage to Create/Delete a distributed application, Average Time/CPU Usage/RAM Usage to Create/Delete a distributed application, Average Time to Peer two clusters, Average Latency between clusters, Average Bandwidth between clusters	

3.2.2 Tests, data collection and analysis

This section presents the preliminary tests performed to evaluate the Low Level Orchestrator, the relevant data collected and first results. Moreover, the infrastructural details of the testbed used will be described in this section.

- The low-level orchestrator functional capabilities regarding the dynamic deployment of Kubernetes clusters and deployment of application components were already demonstrated in the EUCNC & 6G Summit 2023, during the booth exhibition through live demonstrations. This included the integration with the CHARITY AMF component, the dynamic creation of clusters and links between them and the deployment of applications. A preliminary Prometheus-based deployment was used to monitor, collect, and show metrics about the LLO clusters, applications and links operations (i.e., number of clusters running, number of peered clusters, number of applications running, number of individual application components running, integrated *Kube-State Metrics* and *Liqo* metrics) showcased in Figure 1. CloudSigma provided the resources to run the infrastructure hosting the required components. The tests were successful and positive feedback was received from the target audience.

Figure 1 - Orchestrator Metrics Dashboard showcased at EUCnC 2023

- Publication entitled *"Cross-Cluster Networking to Support Extended Reality Services"* submitted to IEEE Internet of Things Journal, where we test a distributed streaming service scenario deployed using the low level orchestrator, leveraging Liqo for the inter-connectivity between clusters and components, analysing its overhead. This includes an evaluation of performance of the Cluster API for creating different types of clusters (both kubeadm and k3s) and different node sizes. Moreover, we also performed an evaluation of performance of Liqo for cluster peering and application offloading. The experimental scenario presented focuses mainly on the overhead of both tools, the time which takes to create clusters and the latency and resource overhead of a peering through Liqo. Using CloudSigma testbed, we were able to achieve cluster creation times of 150 to 270 seconds. The comparison between a single cluster scenario and two-cluster scenario for the use case of a video streaming application, Liqo showed minimal overhead (<500ms).
- Two publications entitled *"Intelligent Multi-Domain Edge Orchestration for Highly Distributed Immersive Services: An Immersive Virtual Touring Use Case"* submitted to 2023 IEEE International Conference on Edge Computing and Communications (EDGE) and *"Towards Establishing Intelligent Multi-Domain Edge Orchestration for Highly Distributed Immersive Services: A Virtual Touring Use Case"* (extended version) submitted to Cluster Computing - The Journal of Networks, Software Tools and Applications, both already accepted. These tests focused on the integration of the Virtual Tour Creator UC, such as the creation of a multi-cluster distributed scenario, consisting of two-clusters which hosted the components composing the use-case distributed evenly across the two-clusters. We collected the feedback of the UC owner which indicated the application ran as expected. Hence, we conclude this test as successful.

3.3 Monitoring Framework (PLEX)

3.3.1 Description, procedure, metrics

Table 9. Description of evaluation subtopic - Monitoring Framework

Subtopic Title: Monitoring Framework	**Partners:** PLEXUS
Short description and evaluation scope: The monitoring framework is evaluated at a functional level to validate its correct behaviour in the collection of monitoring data. Also, in the processing of this data to provide the HLO with the information that triggers the dynamic adaptation of the components susceptible to performance failures. At a performance level, the data update frequency is evaluated, which is managed by the monitoring framework, responsible for providing CHARITY with the most up-to-date data regarding the performance and resource consumption of the deployed components.	
Components involved: Monitoring agents, Monitoring Manager, Resource Indexing	
Where are data collected and stored – measurement points: The testbed for the Monitoring Framework is provided by CloudSigma: a multi-cluster, multi-domain architecture with networking monitoring data exposed by Liqo.	
When are data collected? The performance evaluation script compares data during an hour.	
Instruments/tools: *which modules or questionnaire* Postman and Grafana for functional evaluation of the monitoring framework. Python 3 for performance evaluation.	
Methodology/Procedure: *in which way data are collected* The testing script queries the monitoring agents and the Resource Indexing for an hour and collects the timestamp, latency and response of the requests.	
Metrics to analyze the results Average difference between the instant of the request and the timestamp of the data requested from the monitoring agent. Average latency of the Resource Indexing. Average latency of the Monitoring Manager.	

3.3.2 Tests, data collection and analysis

The monitoring framework test performs requests and stores data during an hour to evaluate:

- Resource Indexing: the ability of the Resource Indexing to offer in real time the most up-to-date values regarding cluster performance and the latency of these responses. The requests analyzed for this component are:
  - Get Resource Status of a single datacenter.
  - Get Resource Status of all the datacenters.
- Monitoring Manager: its ability to respond to requests and the latencies of each of them. The requests analyzed for this component are:
  - Get Metrics History: It returns the monitoring values for metrics on which a performance limit has been set.
  - Get Info: it returns the active alarms and alerts for the application indicated in the request.
- Monitoring agents: The Prometheus servers are tested regarding its ability to gather information and respond with reliable application performance data. To evaluate its competence by informing the rest of the components of Charity's architecture, performance data of metrics that are being monitored are requested for one hour and stored associated with two timestamps, the moment in which the request was made and the moment in which that data was collected by Prometheus.

The monitoring framework test runs in three different scenarios depending on the state of the architecture:

- Stable monitoring scenario: The application components are deployed in the same cluster during test execution time.
- Migration scenario: A monitored metric reaches a limit and the component is migrated to another cluster to avoid performance failures.
- New cluster scenario: A new cluster is deployed so it begins to be monitored by the agent of the cluster and these data are incorporated into the Resource Indexing to gather in their responses the resources available in the expanded architecture.

3.4 Forecasting Model (HUA)

3.4.1 Description, procedure, metrics

Table 10. Description of evaluation subtopic - Forecasting model

Subtopic Title: Forecasting Model	**Partners:** HUA
Short description and evaluation scope: The Forecasting model was evaluated based on its impact in the frame of proactive horizontal scaling in the context of providing reduced latency and advanced fault tolerance, when compared against a standard reactive approach.	
Components involved: Forecasting Component	
Where are data collected and stored – measurement points: In order to gather the resource utilization metrics, Orbital Knight built a testbed, in which 32 simulated players (bots) were simulated on a separate machine and connected to the game server. The game server has been equipped with an additional dedicated module that allows the collecting of the necessary data: the current percentage of the server CPU load (%), the current percentage of the server memory usage (%), the amount of data received and sent over the network (bytes/sec).	
When are data collected? Data collection ran continuously for around 4.5 hours, with 2 seconds intervals, while the created simulation (based on the aforementioned data collection process) lasted 4 days.	
Instruments/tools: Python 3 and the CloudsimPlus simulation framework.	
Methodology/Procedure: Data that correspond to resource demand was collected from ORBK's gaming use-case and leveraged to generated a simulated scenario using the CloudsimPlus simulation framework.	
Metrics to analyze the results: Reduced Latency: Tail latency, Average Execution time, Standard Execution time, Maximum Execution time, Skewness of Execution time, Kurtosis of Execution time. Advanced Fault Tolerance: Mean Time To Failure, Mean Time To Repair, Reliability, Maintainability.	

3.4.2 Tests, data collection and analysis

Table 11 illustrates the experimental results obtained from comparing the suggested proactive horizontal scaling approach with the conventional reactive method in terms of various latency-related metrics. The experimental results showcase that the proposed proactive approach surpasses the reactive one across all examined metrics.

Table 11. Experimental comparison between the proposed proactive horizontal scaling approach and the standard reactive one. Timings are in seconds.

Autoscaling Method	Tail Latency	Avg. Ex. Time.	Std. Ex. Time.	Median Ex. Time	Num. Tasks	Max, Ex. Time	Skewness Ex. Time	Kurtosis Ex. Time
Reactive	5.610	1.767	1.944	1.539	1624817	47.849	9.385	129.665

Intelligent	5.060	1.525	1.123	1.260	1624735	18.909	3.513	20.166

Table 12 illustrates the experimental results obtained from comparing the intelligent proactive (IPFT) approach with the conventional reactive method (RFT) in terms of various fault tolerance-related metrics. The experimental results showcase that the proposed proactive approach surpasses the reactive one across all examined metrics, even when examining various different task scheduling algorithms such as Round-Robin[2], MinMin and MaxMin[3].

Table 12. Experimental comparison in terms of Fault Tolerance

	MTTF	MTTR	Reliability	Maintainability
RFT RR	2.864	19.657	0.741	0.048
IPFT RR	9.506	3.343	0.904	0.230
RFT MinMin	8.733	36.169	0.897	0.026
IPFT MinMin	8.919	5.656	0.899	0.150
RFT MaxMin	3.721	24.239	0.788	0.039
IPFT MaxMin	13.309	7.425	0.930	0.118

3.5 Point Cloud Encoding/Decoding (CNR)

3.5.1 Description, procedure, metrics

Table 13. Description of evaluation subtopic - Point Cloud Encoding/Decoding service

Subtopic Title: Point Cloud Encoding/Decoding service	**Partners:** CNR
Short description and evaluation scope: The Point Cloud Encoding/Decoding (PC E/D) component is used for the fast compression of point clouds. The main intended use is to transmit an huge amount of coloured 3D points.	
Related requirements: No particular requirements. GPU is needed to speed up the performance.	
Components involved: Point Cloud Encoding/Decoding component.	
Where are data collected and stored – measurement points: The data used has been created specifically for test purposes. The test data has been provided by the SRT. It consists in a 3D scene where a person (the Assistant) talks about weather conditions. Another scene where a people is inside a room has been also created and used to test such component (see *Figure 2*). These synthetic scenes are generated in real-time using Unity. The measurements have been conducted using the SRT prototype.	
When are data collected? The test 3D scenes have been created during the Q1-Q2 2023 period. Before this period other scenes have been used to tests the component.	

[2] https://en.wikipedia.org/wiki/Round-robin

[3] https://research.ijcaonline.org/ncetct/number1/NCETCT4017.pdf

Instruments/tools: C++, ffmpeg, GPU shaders
Methodology/Procedure: *in which way data are collected* The PC E/D component is integrated in the UC1-3 Holo Assistant as a software library. The scenes generated by the Unity rendering engine is represented as a set of RGBD images taken from different viewpoints. Since camera calibration is known for each RGBD image, each pixel represent a 3D point with color. This representation of the point cloud permits to the system, taking into account the viewpoint of the user, to generate a set of RGBD images around such viewpoints and transmit them to the Holographic display. The holographic display splats the colored points creating a 3D virtual scene that appears real. The PC E/D is specifically designed to compress and decompress efficiently such RGBD images, so it is a view-dependent compression algorithm.
Metrics to analyze the results Number of 3D points (i.e. resolution of the RGBD images), number of views (i.e. number of RGBD images to compress), Frame-Per-Seconds (FPS).

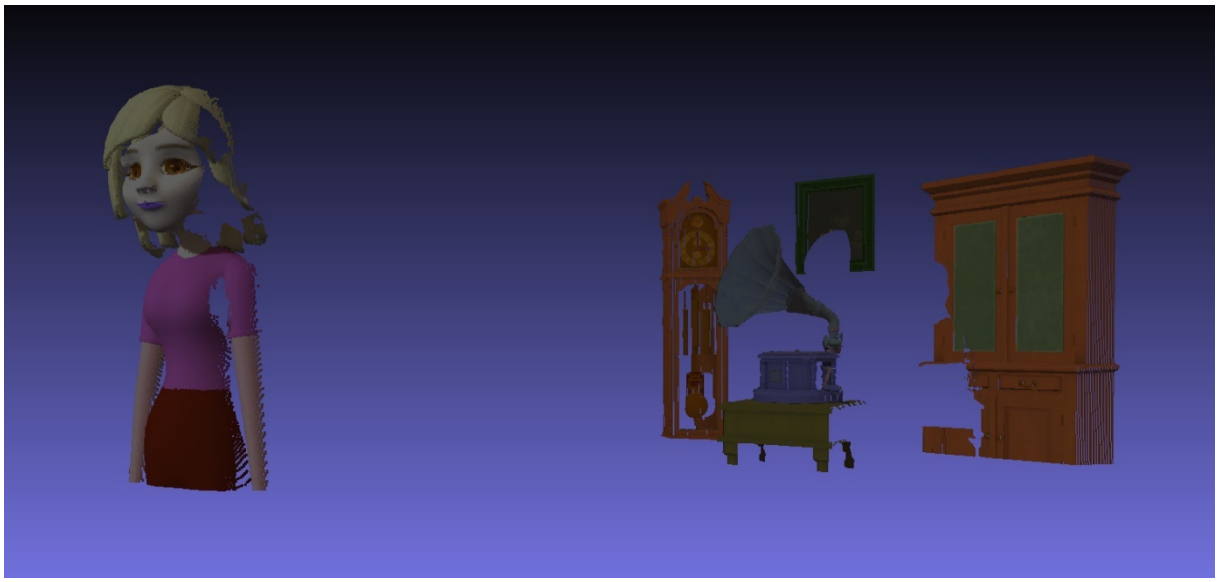

Figure 2. A test scene reconstructed from 8 RGBD views.

3.5.2 Tests, data collection and analysis

The algorithm at the base of the PC E/D component, and the design motivations, are described in detail in the Section 5.6 of the Deliverable 3.2. This component has been tested separately and in the pipeline of the UC1-3. The overall frame rate measured for the test scene is around 5 fps. The resolution of the images are 1280 x 752, 8 scenes at the time are processed and streamed. This number of views is sufficient to permit to the user of the holographic display slightly changes of viewpoint without the need to transmit other data (more details about this point can be found in the D3.2). This performance reported are related to the C++ version. The GPU version is really more performing, but it is not been tested inside the UC1-3 because this requires a tight integration in the image generation pipeline of the UC. Such integration would allow to save computations and memory transfers between the application and the component. Preliminary tests of the GPU version of the component in isolation are really promising; it is reasonable to reach over 30-40 fps. These results have been obtained also on other scenes that are a bit more complex than the assistant test scenes used.

3.5.3 KPIs assessment

Regarding the general objectives of the CHARITY project, this component has been developed in the ambit of the *Objective #4 – Develop highly interactive and collaborative services and applications*, and in particular it satisfies the *KPI-4.3 Specialized data services support: streaming, rendering, compression, caching and encoding*. The performance obtained by the CPU version, and the one potentially obtained by the GPU version (tested in isolation) satisfies the speed performance required by the Holographic Assistant application to reach a good QoE.

3.6 Mesh Merger (CNR)

3.6.1 Description, procedure, metrics

Table 14. Description of evaluation subtopic - Mesh Merger service

Subtopic Title: Mesh Merger service	**Partners:** CNR
Short description and evaluation scope: The Mesh Merger service is a XR data service to assemble together pieces of geometry of an indoor environment to set up a virtual environment for AR applications. The same service, with slightly modifications, can be used to update the virtual environment according to the changes of the real environment. The pieces of geometry are assembled in a mesh called mesh collider, since it is used to resolve collisions enabling the interaction of the virtual objects with the real environments.	
Related requirements: No particular requirement.	
Components involved: Mesh Merger, Game Server (UC3-1)	
Where are data collected and stored – measurement points: The data about indoor environment are collected on-the-fly through a test applications developed by the ORBK which allows to scan a part of the environment using a smartphone equipped with a Lidar. Different mesh colliders of different indoor environments have been created.	
When are data collected? The Mesh Merger has been tested with different acquired single mesh colliders during the Q2-Q3 2023 period.	
Instruments/tools: C++, ARKit[4], TEASER++[5], OpenVDB[6]	
Methodology/Procedure: The methodology for the test procedure is the following. It has been evaluated the time to transmit the pieces of geometry, i.e. the single mesh colliders, provided by the ARKit on the smartphone device equipped with the Lidar, the processing time for the alignment, and the processing time for the fusion of the aligned mesh to create the final mesh collider. First, this evaluation has regarded the component with the data acquired by an application developed by the ORBK. Then, the Mesh Merger component has been turned into a service, based on a REST-API. This service can receive requests of fuse a new single mesh collider into the current mesh collider of the indoor environment. Also in this case the processing time and the transmission time have been evaluated.	
Metrics to analyze the results Data transmission time, processing time, quality visual inspection.	

[4] https://developer.apple.com/augmented-reality/arkit/

[5] https://github.com/MIT-SPARK/TEASER-plusplus

[6] https://www.openvdb.org

3.6.2 Tests, data collection and analysis

The tests conducted are based on real data acquired through an ad hoc application developed by ORBK based on the ARKit. The Mesh Merger service implemented is based on Node.js and follows a REST-API paradigm. The registration and fusion algorithm are based on two open source codes, the TEASER++, for the alignment, and the OpenVDB library, for the fusion of the aligned pieces of geometry into the Mesh Collider, respectively. More details about the alignment and the fusion algorithm can be found in the Section 5.9 of the Deliverable 3.2.

Now, the Mesh Merger service is under the integration in the UC3-1 Collaborative Gaming Application where the Game Server communicates with the Mesh Merger to set up the mesh collider for the game environment. For this Augmented Reality game, the tests conducted demonstrated that the processing time is sufficiently fast to provide to the gamers a good QoE, (i.e., less than 2 seconds are necessary to download and process a new acquisition into the Mesh Collider). In particular, the transmission time is made efficient by using a binary version of a JSON containing a PLY format of the mesh. Even if this data format is not compact, the number of triangles of a single mesh collider is such that it is sufficient for the purpose of an interactive experience and it is easy to manage. The processing is asynchronous, so that multiple users can scan different parts of the indoor environment and set up the game quickly.

3.6.3 KPIs assessment

Regarding the general objectives of the CHARITY project, this component has been developed according to the *Objective #4 – Develop highly interactive and collaborative services and applications*, and in particular it fulfills the *KPI-4.3: Specialized data service support: streaming, rendering, compression, caching, and encoding*.

3.7 UC1-1 Holographic Concert and UC1-2 Holographic meetings (HOLO3D)

3.7.1 Description, procedure, metrics

Table 15. Description of evaluation subtopic - Holographic Concert and Holographic meetings

Subtopic Title: Holographic Concert and Holographic meetings	**Partners: HOLO3D**
Short description and evaluation scope: Our plan is to measure measure latency and data rate. Since the number of consumers and devices is finite and quite low, the latency is expected to be related to the internet connection, rather than anything else. We need to test and get an idea of the maximum latency that is acceptable while not degrading the QoE (video quality and synchronization)	
Related requirements: F_UC1_01: CHARITY provides Cloud server with resources necessary to achieve KPIs. F_UC1_02: CHARITY provides cloud-based software to receive, decompress and render / modify the content in the cloud in real time. F_UC1_03: CHARITY software renders in real time several types of pre-set video modes and resolutions, for several types of Holographic Displays. F_UC1_04: APPLICATION PROVIDER provides speaker PC, video camera, lights, black background, secondary screen. F_UC1_05: APPLICATION PROVIDER provides speaker PC with software to retrieve the raw, 2D video from the video camera and send it to the Cloud server. F_UC1_06: APPLICATION PROVIDER provides client PC, Holographic Display, webcam, mic for 2-way communication with the Speaker PC. F_UC1_07: APPLICATION PROVIDER provides client PC with software to send live video/sound stream to the	

Cloud server. F_UC1_08: APPLICATION PROVIDER provides client PC with software to receive the scrambled, 3D adapted video from the Cloud Server and send it to the Holographic Display. F_UC1_09: APPLICATION PROVIDER provides client PC with software to synchronize with the other connected client PCs. F_UC1_10: APPLICATION PROVIDER, the software F_UC2_08 can choose to retrieve a different type of scrambled, 3D adapted stream from the Cloud server according to the connected Holographic Display. F_UC1_11: APPLICATION PROVIDER provides speaker PC with software to convert the shared content (jpg, pdf, doc, ppt, mp4) to the same type of raw,2d video as in F_UC1_05 (Holographic meetings scenario). NF_UC1_01: The video resolution should be > than full HD (1920x1080) @ 30 fps. NF_UC1_02: Average latency between receiving the raw, 2D video stream from Speaker PC and rendering it for the specific Holo Display resolution and format required by the Client PC<= 30-600 Seconds NF_UC1_03: Average latency between receiving the raw, 2D video stream from Speaker PC and rendering it for the specific Holographic Display resolution and format required by the Client PC<=1000ms (second scenario).
Components involved: cyango-backend, cyango-media-server,cyango-cloud-editor
Where are data collected and stored – measurement points: Data is directly computed in the Speaker and Client PCs. We do not intend store any data.
When are data collected? *i.e., How many days are requested to collect data* Several sessions of 60 minutes each.
Instruments/tools: *which modules or questionnaire* Full hd/4k cameras that support RTMP streaming,
Methodology/Procedure: *in which way data are collected* Data is directly computed in the Speaker and Client PCs
Metrics to analyze the results Available Incoming Bitrate Available Outgoing Bitrate Bytes Discarded On Send Bytes Received Bytes Sent Current Round Trip Time Total Round Trip Time

3.7.2 Tests, data collection and analysis

Initial tests: Video Streaming over wired local network

Our first tests were relevant to both UC1-1 and UC1-2 use cases.

Our initial tests were with TCP – as expected, the stability was good but latency was high.

Over a 1gb wired connection we had an 5000-7000ms delay between the Musician and the Client PCs, using a 1280x720 video stream @ 25fps and ~3500kbps . The latency was mostly induced by the local video manipulation component that vastly depends on the computer performance.

We have used the most difficult template for the Dreamoc Diamond , 4 sides Holographic device.

We then moved to UDP – local streaming. Some video and error handling optimisations were added, the latency slightly improved to 3000-4000ms with the same 1280x720 video stream @ 25fps and ~3500kbps video stream but we had another issue, the sound was no longer synchronized with the

video stream. We used the same video manipulation template for the most difficult, 4 sided Holographic device.

The results were not relevant since the performance greatly depended on the hardware configuration of the Musician and Client PC. The latency was much higher (up to tenfold) than expected) for a local streaming solution and we concluded that a cloud solution with vastly improved computing power was necessary.

Video Streaming – Cloud Server

We have managed to convert our local streaming app and connect it to the CHARITY Edge Cyango-media-server. We have experimented first with many local and web streaming protocols and most of the tests were not promising, due to the high latency induced. After consultations with our partners, we decided to go with a WebRTC protocol.

The streaming was made during two sessions of 2 hours each , the following results were recorded.

For the sake of consistency we tried to use the same video settings for the stream:

Device :Microsoft LifeCam HD-3000

Resolution: HD (1280x720)

Bitrate (kbps): 2500-3500

Frame rate: 25

Video codec: H264

The results are promising, the latency is now under 1000ms, the sound is now synchronized with the video. These test results refer to the raw, non-edited videos, since the cloud video manipulation component is not yet completed.

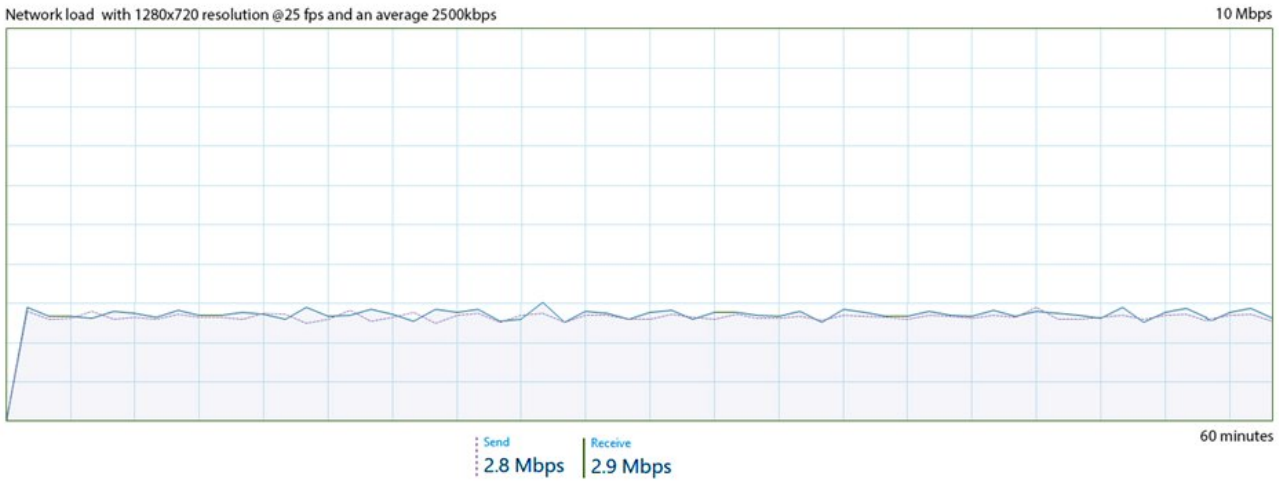

Figure 3. Network load with 1280x720 resolution @25fps and an average 2500kbps

Network load shows a stable average of 2.8 Mbps sent and 2.9 Mbps received on average for an 60 minutes session.

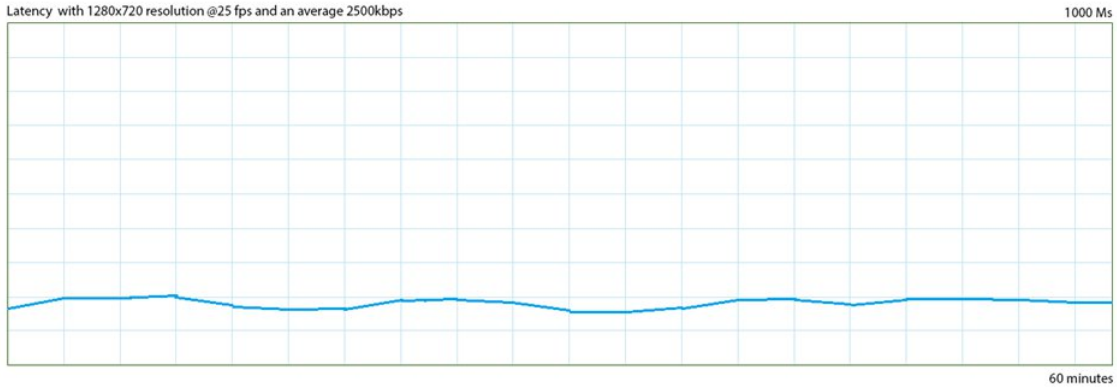

Figure 4. Latency with 1280x720 resolution @25fps and an average 2500kbps

The latency graph shows a relatively stable latency of 150-200ms on average for an 60 minutes session. Using a raw, un-manipulated video stream, this latency is still not good enough for the Holographic Concert Use case, we will continue testing with the cloud video manipulation component, and once completed, we will test again to see the extra latency induced by the video manipulation.

3.7.3 KPIs assessment

KPI-UC1-2.1: Average latency < 20 ms. The average latency is much larger than 20ms, so far it is impossible to say whether this is enough for the Holo Concert use case, further tests are required, especially after the video manipulation and synchronization components are completed.

KPI-UC-1-2:3 Data services required (rendering, compression, caching, encoding) =>4. The tests were made with three different data services (transcoding video stream, rendering video and networking).

3.8 UC1-3 Holographic Assistant (SRT)

3.8.1 Description, procedure, metrics

It is important to note that the information reported for this use case is the final one due to the SRT leaving the consortium.

Table 16. Description of evaluation subtopic - Assistant providing holographic 3D visual and spoken information

Subtopic Title: Assistant providing holographic 3D visual and spoken information	**Partners:** SRT
Short description and evaluation scope: The task of this use case is to develop and implement an ecosystem of software and services to create an assistant, reacting to spoken requests by showing visual 3D holographic information and spoken output as answer. The requirement for enabling holographic 3D is the need for 3D Point Data streamed to the local client system to overcome large EDGE - CLIENT transmission and processing delays since the requirement is that the visual information is generated in an EDGE-system. The use case is evaluated by checking the correct interaction and data exchange between the modules. On client side, frame-rate of the displayed 3D hologram is important and measured. The delay and frame-rate of the streamed 3D Point cloud is also measured, but not so important for providing a good user experience. The time between spoken requests and reaction by the system is another point to be evaluated.	
Related requirements: F_UC1_13: APPLICATION PROVIDER provides Unity software to render the content and assistant. F_UC1_14:APPLICATION PROVIDER receives eye position data and interaction data from client device. F_UC1_15: APPLICATION PROVIDER provides software to generate the 3D point cloud from rendered content frame by frame. F_UC1_17: APPLICATION PROVIDER provides software to model behavior of the holographic assistant.	

F_UC1_18: APPLICATION PROVIDER uses cloud services to process speech – i.e., google cloud text to speech and speech to text. F_UC1_19: APPLICATION PROVIDER uses cloud services to gather information – weather, stocks dependent upon requests from USER. F_UC1_20: APPLICATION PROVIDER uses cloud services to connect to a chat bot for smooth ping pong interactions – i.e., from IBM Watson. F_UC1_21: APPLICATION PROVIDER is able to visualize information. F_UC1_23: APPLICATION PROVIDER receives speech data from client device. F_UC1_24: APPLICATION PROVIDER uses other cloud services to gather information. F_UC1_26: CHARITY provides a software to encode 3D point cloud data (optional using GPU acceleration) into a compressed format capable for network transport and streaming and sends it to client device. F_UC1_27: CHARITY optimizes bandwidth by only transferring differences between 3D point cloud frames. F_UC1_28: CHARITY provides a software running on a client device to decode 3D point cloud data from network data stream. F_UC1_29: APPLICATION PROVIDER provides a software running on a client device to compute received point cloud data dependent from local eye-tracking data and optionally provided occlusion data into holograms. F_UC1_30: APPLICATION PROVIDER records USERs voice and outputs assistant speech. NF_UC1_05: Data services required (raw data streaming, rendering, compression, caching, encoding) >=5. NF_UC1_06: Latency in speech input (human) and speech output (assistant) <= 2 sec.
Components involved: SRT_SW_CLIENT, SRT_SW_CONTENT, SRT_SW_PCGEN, CHARITY_SW_PCENC, CHARITY_SW_PCDEC, SRT_SW_BEHAVIOR, external services
Where are data collected and stored – measurement points: SRT_SW_CLIENT - hologram frame-rate, speech reaction time CHARITY_SW_PCENC - Point cloud frame rate
When are data collected? When the system at all is running. Some measurements are done when user speaks, thus interacts with the assistant.
Instruments/tools: Console output to show current measurements. Preparations to connect to monitoring interfaces were started. Stop watch until spoken request is finished and result appears / plays back.
Methodology/Procedure: Time-Measurement within the software using operating system provided functions.
Metrics to analyze the results Frame-time - measure time between showing a frame and average over time, Frame-rate is 1 / Frame time Time - measure time between a starting event until the expected result is appearing.

3.8.2 Tests, data collection and analysis

The tests made are evaluating how the overall ecosystem reacts and operates. The general test and use-scenario is like the following. The user speaks the wake-word, which is currently “Charity” to activate the assistant and set it into listening mode at SRT_SW_CLIENT. The requirement for this is to be below 2s which is typically achieved. But this strongly depends on the load at google speech services - when using a paid account, this time is always achieved. Then after the assistant is in listening mode, the user may ask a question like “How is the weather in <city>”. This leads to several actions, SRT_SW_BEHAVIOR will receive the request after translated to text by google services, analyses the

context -> weather, fetches information about weather in given city through API at openweathermap.org and provides information to SRT_SW_CONTENT to generate visual information. It then sends the resulting text to google services to create the voice data, which is then played back at SRT_SW_CLIENT. The visual information is generated by SRT_SW_CONTENT in Unity 3D adding text, icons and animated assistant reactions related to the current context. The rendered views are processed into 3D Point clouds frame by frame via SRT_SW_PCGEN and compressed via CHARITY_SW_PCENC. This data sent to the client via TCP / IP is received and decompressed via CHARITY_SW_PCDEC. Finally the required views are generated from this 3D point cloud frame and the hologram is generated and displayed. The latency between providing eye-coordinates and rendering the new views from the 3D Point cloud is always lower than 60 ms. The frame-rate of the streamed 3D Point cloud is in the current implementation about 5 FPS. With GPU-optimizations this can easily reach 30 FPS and more. The delay between sending the 3D Point cloud data and receiving plus decoding on client is about 3-4 seconds, mostly due to time taken for compressing and buffering 3D point cloud data via h264 compression in FFMPEG libraries. But delay this is not so important for assistant scenarios since several seconds delay is expected to get an answer from such an application.

3.8.3 KPIs assessment

KPI-UC-1.1 Average latency between sending input data and receiving 3D- point cloud data <=60ms

This KPI relates to how quick the hologram is updated upon a change of eye-position (thus user moves). This is a local process, where the received 3D point cloud is rendered again according to the updated eye-location. This process was fully implemented in the GPU, the aimed 60 ms delay are always reached.

KPI-UC-1.5 (assistant) Latency in speech input (human) and speech output (avatar) <= 2 sec

This KPI defines the expected latency when speaking with the assistant and when first reaction should occur. We decided to implement a keyword or activation word. In this scenario the reaction is typically below 2 seconds. In some cases dependent on load at the speech recognition service (google S2T) this might take a bit longer. By booking a paid plan, this can be eliminated. In context of this evaluation, a free plan was used, but with changing conditions.

KPI-UC-1.3 Frame rate of the holographic visualization >= 30Hz

The frame-rate of the holographic visualization is basically always above 30Hz. Because this scenario already relates to the rendering of the 3D point cloud according to 60Hz eye-tracker update rate. This KIP is mostly influenced by hologram computation load - for the implemented holographic assistant graphic design the 30Hz are always reached since the fill rate of the scene is typically below 50%. But regarding the frame-rate of the content from EDGE to CLIENT, currently only 5 Hz are possible. This is due to more required optimization of the point cloud generation process - in ideal case this part should run also on GPU. Then it is expected that frame-rate of 3D point cloud generation and compression will reach 30Hz easily.

3.9 UC2-1 VR Medical Training (ORAMA)

3.9.1 Description, procedure, metrics

Table 17. Description of evaluation subtopic - Realistic simulation in VR medical training

Subtopic Title: Realistic simulation in VR medical training	**Partners: ORAMA**
Short description and evaluation scope: ORAMA plans to use the metrics of latency, data rate and number of users in order to determine the maximum latency that is supported in relation to the number of concurrent users in a VR session. The end-to-end latency derives from three factors: processing in the edge /cloud resources, transmission over the network and processing on the HMD. End-to-end latency includes the rendering and the streaming latency, the HMD render time bias for decoding and blit as	

well as the jitter and refers to the time since a user movement is registered by the system and for the corresponding image to be displayed on the headset's screen. In addition, we aim to approximate both the data cost and the latency cost (network related) each additional user adds to a VR session.
Related requirements: **F_UC2_01:** APPLICATION DEVELOPER: Use the mirror networking service or similar for matchmaking, creation of session and selection of an already existing session (IP, location, userid master) photon. **F_UC2_03:** APPLICATION DEVELOPER: Session management through a relay server or message broker in the cloud. **F_UC2_07:** APPLICATION DEVELOPER: The application component running on the HMD should be aware of the connected resources (app instance on edge) where part of the application has been offloaded. **F_UC2_08:** APPLICATION DEVELOPER: The application running on the HMD should be able to connect via standardized protocols to the resources (app instance on edge) where part of the application has been offloaded. **F_UC2_11:** APPLICATION DEVELOPER: Support continuous streaming of two images (one per eye) per user from the edge resource node to the HMD. **F_UC2_13:** APPLICATION DEVELOPER: The resource discovery mechanism of CHARITY should offload part of the application functionality from the HMD to nearby edge resource considering lowest average latency. **F_UC2_17:** APPLICATION DEVELOPER: Establish communication of the HMD and the Remote Service (RS) in the cloud/edge when launching the app on the HMD. **NF_UC2_01:** USER, APPLICATION DEVELOPER: Round trip time (RTT) latency <15ms. **NF_UC2_04:** USER, APPLICATION DEVELOPER: Connectivity from user HMD device <10 ms. **NF_UC2_10:** APPLICATION DEVELOPER: Receive error messages on potential problems with existing resources, continue the VR app by communicating with another newly discovered resource (discovery and placement). **NF_UC2_17:** USER, APPLICATION DEVELOPER: Performed actions from all users must be synchronized to the output rendered image of each individual user's HMD with lowest average latency.
Components involved: LSPart1, LSPart2, CTLInterface, HMDApp, Photon
Where are data collected and stored – measurement points: Data is collected from the LSPart1, directly computed in MAGES SDK or retrieved via the Photon Network API, as well as from the HMD. All data are sent and stored to the HMD and could be further uploaded to the Microsoft Azure cloud.
When are data collected? *i.e., How many days are requested to collect data* Each session of tests lasts approximately 15 minutes, for object movement metrics.
Instruments/tools: *which modules or questionnaire* Mobile HMD, LSPart1, LSPart2, log data.
Methodology/Procedure: *in which way data are collected* Data is captured by Streamer Server instances running on the LSPart1, via a metrics obtainer script, and stored in the LSPart1. Latency metrics are obtained through timestamped packets exchanged with the HMD. The rates for each record (where possible) are computed at runtime and stored in the records file. User input can greatly vary in each VR session. We aim to scale to 50 users in the experiments.
Metrics to analyze the results 1. Round trip time, render time (single), 2. Message count/rate of exchange distinguished as follows: a) Transformation changes recorded by MAGES SDK (position, orientation) b) Total Byte cost sum of logic-level messages (total byte rate, total incoming /outgoing message rate/count) c) Packet count

3.9.2 Tests, data collection and analysis

A number of testing sessions were conducted, in which experiments were incrementally staged to reach a large number of CCUs, exploiting both real users with available HMDs and simulated users via the exploitation of bots. The aim is to reach the target of 50 CCUs by the end of the project. The experiments in this deliverable aimed to assess the Use Case components separately from the orchestration of the CHARITY platform on one hand to assess the deployment of the developed components in one of the testbeds provided by the project and on the other hand to assess the metrics and KPIs before further exploiting the orchestration functionalities of the CHARITY platform. The HMDs for the experiment were provided from ORAMA and the participants were members from the ORAMA team located in Greece. The LSPart1 and LSPart2 components were deployed in the Sweden testbed provided by CloudSigma (CS). For each LSPart1 a VM was created, deployed and a GPU was assigned to it. In addition another VM was exploited for simulating 15 users via bots. The LSPart2 was deployed as a docker image. Each participant's HMD was connected to a separate VM, where the LSPart1 was running. In this setup for each HMD data would be streamed from/to a separate cloud-node (not edge node), exclusive to each HMD and the data were captured by the streamer server instance running on the HMD. The HMD was connected through 5Ghz wi-fi. The following tests were conducted:

a) Test 1 - 1 HMD user and 15 bots

The test was conducted via the deployment of 1 VM (Windows Tiny 10) with GPU for LSPart1 for the HMD user, 1 VM (Windows Tiny 10) for the LSPart1 without graphics acceleration to simulate 15 users and 1 docker image for LSPart 2. The metrics of latency in ms, the frames per second, the packet lost % and the sent rate are reported in the following *Figure 5*. Latency comprises of individual metrics for econding, decoding, send latency and RTT as depicted in different colors. The RTT latency is in the range of 50 ms (red part in the latency figure), which is justified with the HMD user being located in Greece and connected to the LSPart1 component deployed in the Sweden testbed. Still both the encoding and decoding latency are below 20ms. The average framerate is at 60-80 fps, the % of packet lost is below 5% and the sent rate is in the range of 80-110 Mbps.

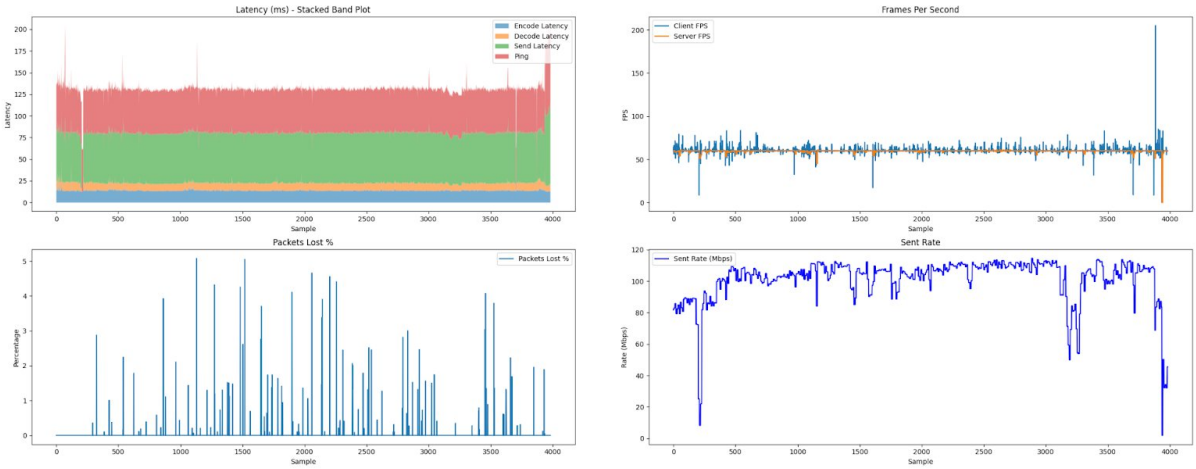

Figure 5. Metrics from Test 1. The latency in ms (upper left), the frames per second (upper right), the packet lost % (lower left) and the sent rate (lower right).

b) Test2 - 2 HMD users and 22 bots

The test was conducted via the deployment of 2 VM (Windows Tiny 10) with GPU for LSPart1 for the 2 HMD users, 2 VM (Windows Tiny 10) for the LSPart1 without graphics acceleration to simulate 22 users (11 for each) and 1 docker image for LSPart 2. The metrics of latency in ms, the frames per second, the packet lost % and the sent rate are reported in the following *Figure 6*. The metrics in Test 2 did not exhibit major differences to those of Test 1.

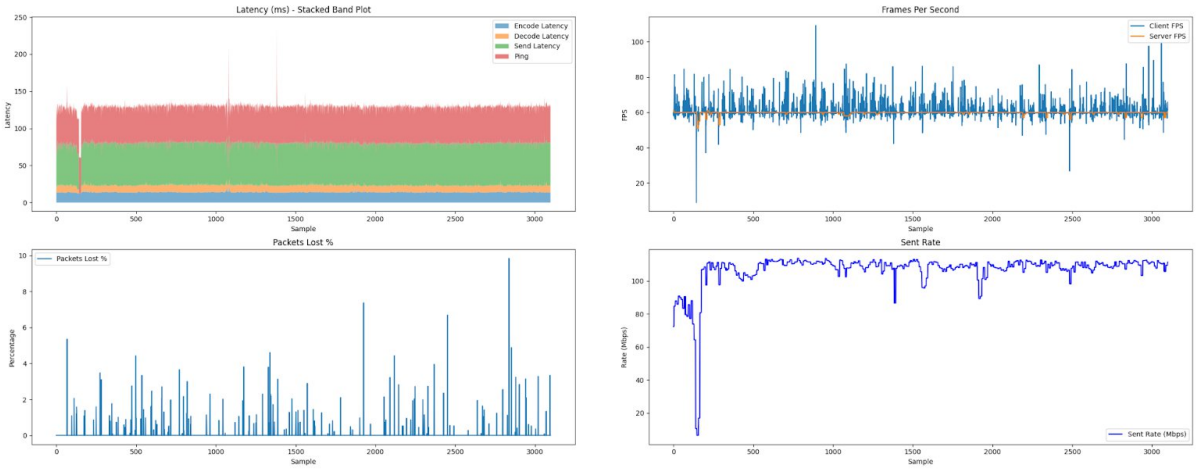

Figure 6. Metrics from Test 2. The latency in ms (upper left), the frames per second (upper right), the packet lost % (lower left) and the sent rate (lower right).

c) Test3 - 4 HMD users and 20 bots

The test was conducted via the deployment of 5 VM (Windows Tiny 10) with GPU for LSPart1 for the 2 HMD users, 2 VM (Windows Tiny 10) for the LSPart1 without graphics acceleration to simulate 20 users (10 for each) and 1 docker image for LSPart 2. Figure 7 depicts the actual VR scene with the 24 CCUs. The metrics of latency in ms, the frames per second, the packet lost % and the sent rate are reported in *Figure 8*.

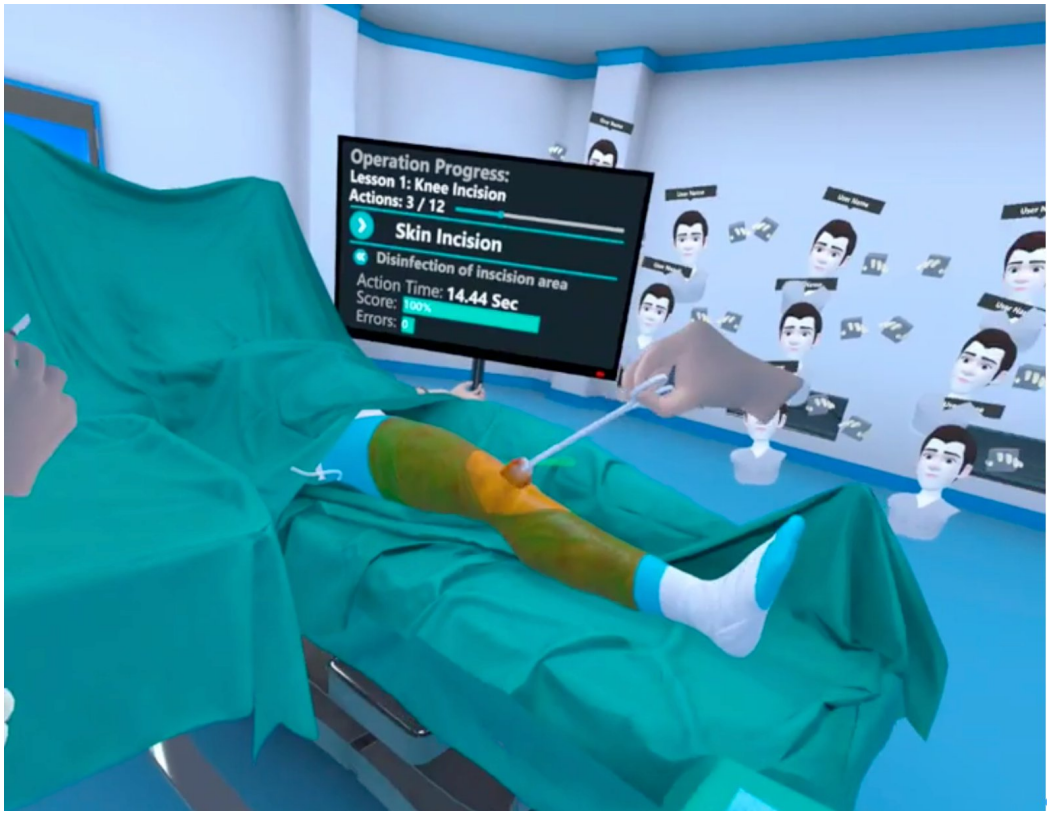

Figure 7: Test3 with 4 HMD users and 20 bots

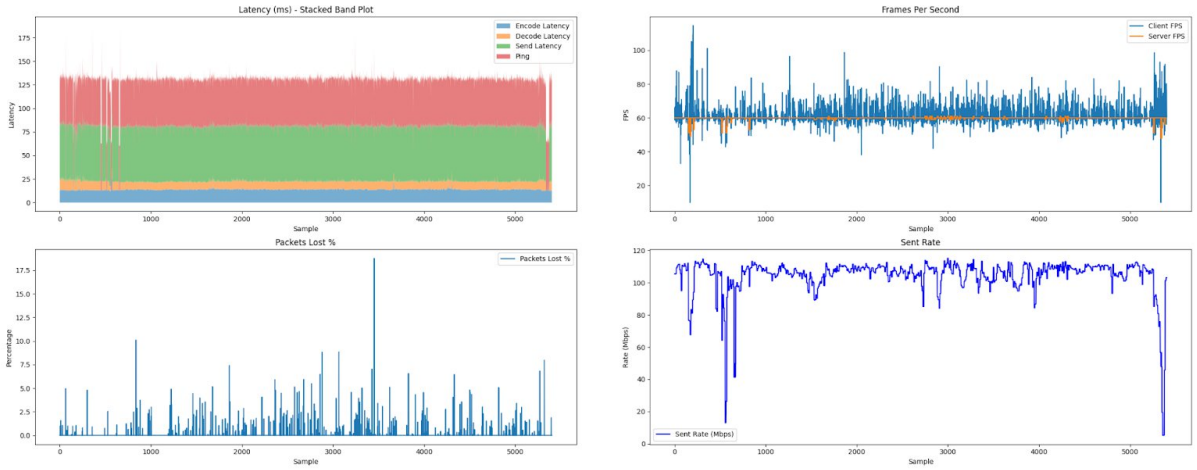

Figure 8. Metrics from Test 3. The latency in ms (upper left), the frames per second (upper right), the packet lost % (lower left) and the sent rate (lower right).

3.9.3 KPIs assessment

KPI-UC2.1: Average latency < 20 ms. In all above experiments the encode/decode latency is below 20 ms. The higher figures in sum of the different latencies are due to the fact that the HMD users were located in Greece and connected to the LSPart 1 component deployed in the Sweden testbed.

KPI-UC-2.2: Number if CCU >50). This KPI was partly reached with 24 users (4 HMDs and 20 bots) , still a higher number of users was reached than initially anticipated in D4.2 (10 users)

KPI-UC-2.3: Number of different VR HMDs >5. In this deliverable we conducted tests with 4 HMDs and aim to increase in terms of number in the next round of experiments.

KPI-UC-2.4: Data services required (rendering, compression, caching, encoding) =>4. The experiments were conducted three different data services (rendering, physics, networking) and the additional service for communication.

KPI-UC-2.5: Automated configurable soft-body simulation for objects with large number of vertices >= 8.000 vertices. The medical sample app from ORAMA, which serves as the pilot prototype for testing the developed services and exploiting the functionalities of the CHARITY platform, includes different objects with varying numbers of vertices (e.g., the patient's leg with 8252 vertices), enabling configurable soft-body simulation.

3.10 UC2-2 VR Tour Creator (DOTES)

3.10.1 Description, procedure, metrics

Table 18. Description of evaluation subtopic - Virtual Experiences Builder for the web

Subtopic Title: Virtual Experiences Builder for the web	**Partners: DOTES**
Short description and evaluation scope: We plan to measure latency, data rate and number of consumers to understand the overload limits and the maximum latency that is acceptable while not degrading the QoE of the number of concurrent users and the number of requests from the client side.	
Related requirements: F_UC2_22: APPLICATION DEVELOPER: Cloud video editor. Allows USER to edit the video file on the cloud with tools like trim, transitions and color grading.	

F_UC2_ 23: APPLICATION DEVELOPER: Real-time video streaming. The VIEWER must be able to consume the live streaming video on the Story Front-end. F_UC2_24: APPLICATION DEVELOPER Real-time 3D Model server-side render. The VIEWER should be able to see the 3D model with adaptative quality depending on the network quality. F_UC2_25: APPLICATION DEVELOPER: Real-time audio translation. The edge cloud should be able to process the audio of a live streaming video and transcribe it on the APPLICATION DEVELOPER: Cloud processing power. The edge cloud should have enough resource allocation depending on the demand of the media files that the VIEWER requests. F_UC2_26: APPLICATION DEVELOPER: Cloud video editor. Allows USER to edit the video file on the cloud with tools like trim, transitions and color grading. NF_UC2_19: USER, APPLICATION DEVELOPER: Data rate >50 Mbps supported by at least 5Ghz wifi or 5G. NF_UC2_20: APPLICATION DEVELOPER: Receive error messages on potential problems with existing resources, continue the VR app by communicating with another newly discovered resource (discovery and placement). NF_UC2_21: USER: Continue using the application in case of problems in network resources with minimal delay. NF_UC2_22: ADMINISTRATOR: Maintain application integrity and user's security. NF_UC2_23: APPLICATION DEVELOPER, USER: Proximity of the relay server based on the users' footprints. NF_UC2_24: APPLICATION DEVELOPER: GPU and CUDA acceleration capabilities available at edge nodes, where part of the application is instantiated.
Components involved: cyango-story, cyango-backend- cyango-database, cyango-worker- cyango-media-server,cyango-cloud-editor
Where are data collected and stored – measurement points: Data is collected on the cyango-story or cyango-cloud-editor and stored on the Prometheus instance
When are data collected? The amount of time to collect the data depends on usage of the end-user, but typically it can be around 5 hours.
Instruments/tools: 360 Cameras that support RTMP streaming, cyango-story, cyango-backend- cyango-database, cyango-worker-cyango-media-server, cyango-cloud-editor, HMDs
Methodology/Procedure: Data is directly computed in the cyango-story or cyango-cloud-editor on the client's browser side, that can be desktop, mobile or HMD. The data is sent to cyango-backend and then exposed and stored via an endpoint to a Prometheus instance.
Metrics to analyze the results • Available Incoming Bitrate; • Available Outgoing Bitrate; • Bytes Discarded On Send; • Bytes Received; • Bytes Sent; • Current Round Trip Time; • Total Round Trip Time;

3.10.2 Tests, data collection and analysis

VR Video Livestreaming

The livestreaming use case was implemented on the cyango-media-server component which is hosted in the edge. There were many experiments and tests to achieve a working prototype of a real time 360 video experience. This was achieved by using the WebRTC protocol, after many unsuccessful tests

with HLS, LHLS and DASH, which are also streaming protocols, but with too much delay.

We implemented the Livestreaming VR funcionality with a metrics middleware to evaluate a set of bandwidth and video conditions params. We made a test by streaming a 5.7k video to the cyango-media-server and consuming it via the cyango-story component on different premises. The streaming was made during 2 hours, which we retrieved some metrics. Some of the most relevant metrics gathered are shown in the *Figure 9 - Figure 13* below:

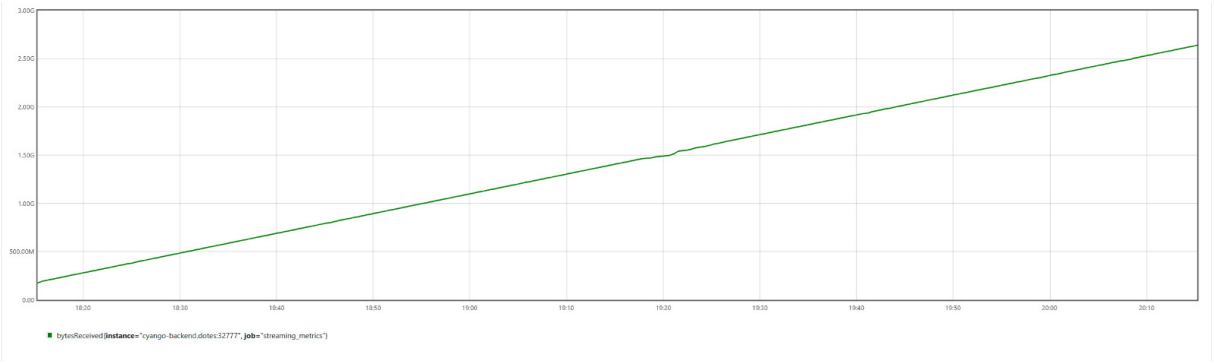

Figure 9. Bytes received

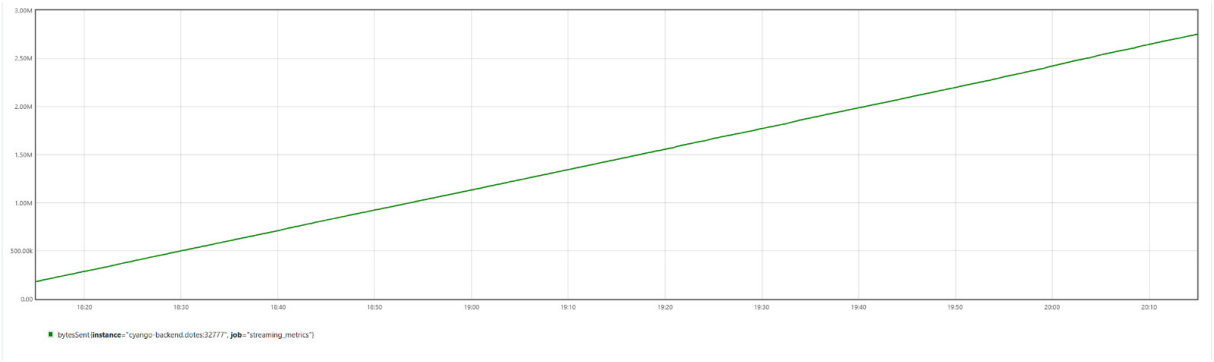

Figure 10. Bytes sent

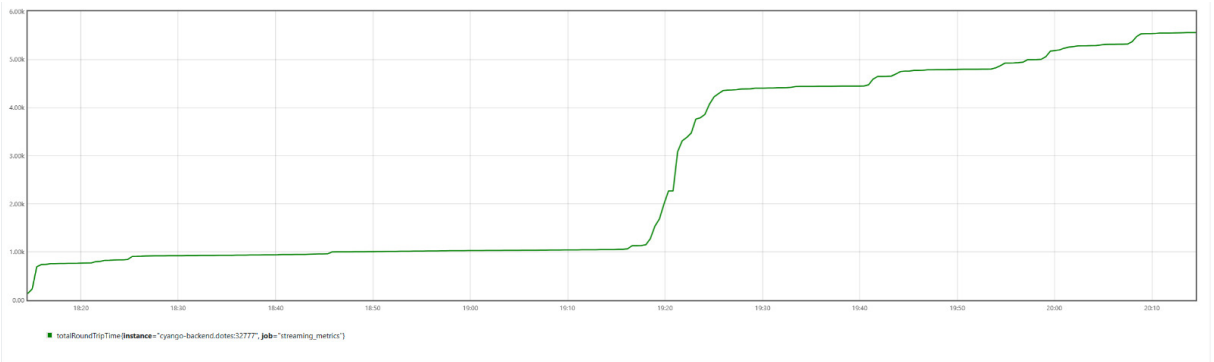

Figure 11. Total Round Trip

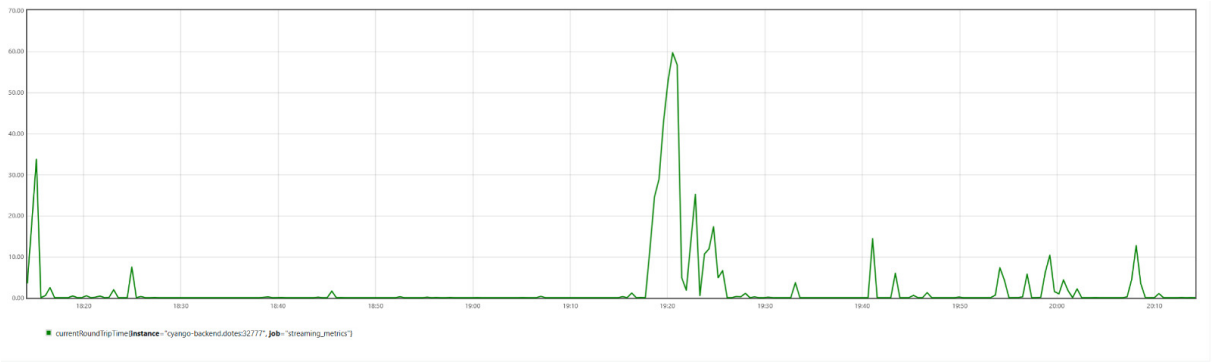

*Figure 12.*Current Round Trip

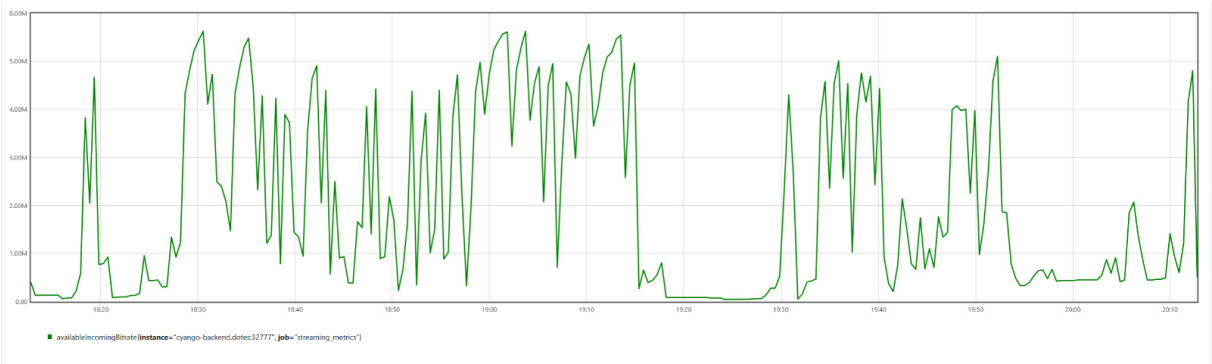

Figure 13. Available incoming bitrate

Video Converter

The video converting use case has the most progress in terms of testing and implementation. This feature requires a continuous improvement and trials with multiple video formats and sizes. We want to enable our users to upload any kind of video and stream it in the most effective way. The custom algorithm we implemented derives from the cyango-worker component and it is responsible for receiving the uploaded file and convert it using the ffmpeg library. The algorithm uses a encoding ladder strategy that generates many quality levels of that file (video, image and audio) allowing a better experience. We also found out that there are limitations on the video quality that can be played in the VR headsets, mainly about the bitrate and resolutions. To achieve a good balance between quality and performance for any kind of video is hard, because there are many factors and fine-tuning parameters that can improve or degrade the user experience. We made tests with many video formats and resolutions with different bitrates for multiple combinations of ffmpeg commands. Until now we achieved a considerate good relation between quality, size and bitrate that converts any video format up until 500 Mb to a streamable HLS playlist . But many improvements are needed.

We also implemented an analytics architecture to store metrics about file converting performances and livestreaming video 360 latency and other related metrics.

3.10.3 KPIs assessment

KPI-UC2.1: Average latency < 20 ms. The average latency is difficult to measure, although along the tests we've made, and knowing all the external factors that could affect the latency, the average latency can be less than 20 ms in optimal conditions.

KPI-UC-2.2: Number if CCU >50). The number of concurrent users is still not deeply tested, mostly because of the cloud resources. Until now with the existing cloud resources we could have only 2 concurrent users transcoding/streaming video, and this is already considering a sufficient quality of experience.

KPI-UC-2.3: Number of different VR HMDs >5. Streaming video we could only test with 1 VR HMD until

now, but planning to test the VR video streaming with 5 concurrent users with VR HMD soon.

KPI-UC-2.4: Data services required (rendering, compression, caching, encoding) =>4 The tests were made with three different data services (transcoding VR video, rendering VR video and 3D and networking).

3.11 UC3-1 Collaborative Gaming (ORBK)

3.11.1 Description, procedure, metrics

Table 19. Description of evaluation subtopic - Mobile multiplayer game utilising AR technology

Subtopic Title: Mobile multiplayer game utilising AR technology	**Partners:** ORBK
Short description and evaluation scope: UC 3.1 will measure RTT and latencies between all core UC components: Game Clients, Game Servers and Mesh Merger. Game Servers Manager will be an intermediary element facilitating communication across UC components. ORBK role involves also testing the efficient management of Docker images for game server deployment within the CHARITY platform. This includes tasks such as image uploading, updates, and deployment. We will assess the user-friendliness and effectiveness of the CHARITY management website's and deployment API for simplicity and reliability.	
Related requirements: F_UC3_1 ADMINISTRATOR: CHARITY should have a repository which will store docker images. F_UC3_2 ADMINISTRATOR: CHARITY should have a website (CHARITY management website) which can be used to update docker images to docker images repository. F_UC3_3 ADMINISTRATOR: CHARITY management website should display docker images uploaded to docker images repository. F_UC3_4 ADMINISTRATOR: CHARITY management website should display deployed docker images status. F_UC3_5 APPLICATION DEVELOPER: CHARITY must have a deployment API which can be used to request for a new game server instance. F_UC3_6 APPLICATION DEVELOPER: CHARITY must be able to deploy docker images. F_UC3_7 APPLICATION DEVELOPER: Deployment API must return host public IP after deploying docker image. F_UC3_8 APPLICATION DEVELOPER: Deployed docker image must be reachable by UDP protocol through one of predefined ports. F_UC3_11 APPLICATION DEVELOPER: CHARITY must deploy docker image as close (geolocation) to requesting player as possible (with lowest latency). F_UC3_12 ADMINISTRATOR: CHARITY should monitor deployed docker image status (CPU usage, RAM usage, overall performance).	
Components involved: Game Clients (iOS app), Game Server (Docker image), Mesh Merger (Docker image), Game Servers Managers (deployed outside CHARITY platform)	
Where are data collected and stored – measurement points: The latency data will be stored in a DB managed by Game Servers Manager. GSM will also expose the latency data to CHARITY's monitoring services in order to be able to receive alarms and alerts and react on them.	
When are data collected? The data will be collected during the runtime of the game. Whenever at least one Game Server is deployed and running the data will be collected, exposed and analysed.	
Instruments/tools: Game Servers manager, Game Server and Game Clients will use build-it tools to measure latencies.	

Methodology/Procedure: *in which way data are collected*
The latencies we will measure will also include the time required for computation of the answer that will be sent to the asking components. This means that in case of Game Server it will include time required to perform full game simulation and in case of Mesh Merger it will include time to perform mesh merging operations.
Metrics to analyze the results – Latency between Game Clients and Game Server – Latency between Game Server and Mesh Merger

3.11.2 Tests, data collection and analysis

The components involved include Game Clients (iOS app), Game Server (Docker image), Mesh Merger (Docker image), and Game Servers Managers deployed externally to the CHARITY platform. We configured and prepared the Game Server Docker image for manual deployment within the project's infrastructure. Manual deployment was successful and we were able to test connections established between Game Clients and Game Servers. Game Servers Manager is now being developed and prepared for fully automatic deployment of the Game Servers. We are currently testing CHARITY AMF API connectivity and functions. As soon as the orchestration components will be operational we will perform further planned tests.

Latency data will be collected and stored in a database managed by the Game Servers Manager (GSM). GSM will also share this data with CHARITY's monitoring services to receive alerts. Data collection will occur during game runtime, triggered when at least one Game Server is running.

Built-in tools within Game Servers Manager, Game Server, and Game Clients will be used to measure latency. Our methodology includes measuring latency, which covers the time for responses to be computed. For instance, in the Game Server, it includes game simulation time, while in the Mesh Merger, it covers mesh merging time.

We'll analyse key metrics, including latency between Game Clients and Game Server and latency between the Game Server and Mesh Merger. Our focus remains on comprehensive testing, systematic data collection, and detailed analysis to ensure the robustness of our system.

```
update_timestamp “2023-10-31T10:44:04.345”

#network latency between Game Clients and Game Servers

game_server_client_network_latency_avg{game_server="45.67.89.123"} 30
game_server_client_network_latency_avg{game_server="136.142.15.88"} 35

game_server_client_network_latency_min{game_server="45.67.89.123"} 10
game_server_client_network_latency_min{game_server="136.142.15.88"} 15

game_server_client_network_latency_max{game_server="45.67.89.123"} 50
game_server_client_network_latency_max{game_server="136.142.15.88"} 55

#network latency between Game Servers and Mesh Mergers

game_server_mesh_merger_network_latency_avg{game_server="45.67.89.123", mesh_merger="203.120.58.79"} 40
game_server_mesh_merger_network_latency_avg{game_server="136.142.15.88", mesh_merger="182.74.25.38"} 45

game_server_mesh_merger_network_latency_min{game_server="45.67.89.123", mesh_merger="203.120.58.79"} 20
game_server_mesh_merger_network_latency_min{game_server="136.142.15.88", mesh_merger="182.74.25.38"} 25

game_server_mesh_merger_network_latency_max{game_server="45.67.89.123", mesh_merger="203.120.58.79"} 60
game_server_mesh_merger_network_latency_max{game_server="136.142.15.88", mesh_merger="182.74.25.38"} 65
```

Figure 14. Measured network latency data: Game Clients <-> Game Servers and Game Servers <-> Mesh Mergers

```
#computational latency between Game Clients and Game Servers

game_server_client_computational_latency_avg{game_server="45.67.89.123"} 130
game_server_client_computational_latency_avg{game_server="136.142.15.88"} 135

game_server_client_computational_latency_min{game_server="45.67.89.123"} 110
game_server_client_computational_latency_min{game_server="136.142.15.88"} 115

game_server_client_computational_latency_max{game_server="45.67.89.123"} 150
game_server_client_computational_latency_max{game_server="136.142.15.88"} 155

#computational latency between Game Servers and Mesh Mergers

game_server_mesh_merger_computational_latency_avg{game_server="45.67.89.123", mesh_merger="203.120.58.79"} 240
game_server_mesh_merger_computational_latency_avg{game_server="136.142.15.88", mesh_merger="182.74.25.38"} 245

game_server_mesh_merger_computational_latency_min{game_server="45.67.89.123", mesh_merger="203.120.58.79"} 220
game_server_mesh_merger_computational_latency_min{game_server="136.142.15.88", mesh_merger="182.74.25.38"} 225

game_server_mesh_merger_computational_latency_max{game_server="45.67.89.123", mesh_merger="203.120.58.79"} 260
game_server_mesh_merger_computational_latency_max{game_server="136.142.15.88", mesh_merger="182.74.25.38"} 265
```

Figure 15. Measured computational latency data (RTT): Game Clients <-> Game Servers and Game Servers <-> Mesh Mergers

3.11.3 KPIs assessment

KPI-UC-3.1: RTT (gaming) sum of network latency and game server response time < 100ms

At this point, we have made substantial progress toward achieving KPI-UC-3.1. The sum of network latency and game server response time is well within our target, currently standing at less than 100ms, indicating a highly responsive gaming environment.

3.12 UC3-2 Manned-Unmanned Operation Trainer (UTRC)

3.12.1 Description, procedure, metrics

Table 20. Description of evaluation subtopic - Cloud Native Flight Simulator

Subtopic Title: Cloud Native Flight Simulator	**Partners:** Collins Aerospace
Short description and evaluation scope: We seek to observe and measure the performance of the use case - in particular latency, frame rate, resolution and rendering features. We also seek to demonstrate scalability to multiple users	
Related requirements: F_UC3_13: The simulation must facilitate collaboration between users to efficiently execute the simulated mission F_UC3_14: Scenery generation may support scenery with different weather F_UC3_15: The simulated environment should allow participants to join or leave simulation at any time F_UC3_16: The simulation should enable prediction of background scenery demands so that it can be pre-fetched by any component from off-line storage F_UC3_17: The simulation should enable custom tiling of cloud-based image generator output to facilitate variable resolution across a single frame NF_UC3_18: The simulation should adapt imagery frame rate and resolution in accordance with available	

bandwidth, observed latency, and user equipment capabilities. NF_UC3_20: The RTT from user action to presentation of updated imagery should be < 15ms NF_UC3_21: Number of concurrent users (virtual & real) in a single simulation scenario should be > 30 F_UC3_22: The simulation should be able support both active participants (present in the simulated environment) and passive observers (not present in the simulate environment) NF_UC3_23: The video resolution of presented imagery must be greater than 60 FPS 4K. NF_UC3_11: The simulated environment must provide a consistent simulation state across all users, including rendering of other user activities
Components involved: Cloud services (Image Generator, Virtual Frame buffer, Transcoder, Media server, Session Manager) and Edge Services (Cache, stream receiver, upscalers, flight oracle, streamsender), Kubernetes, Prometheus
Where are data collected and stored – measurement points: Data is collected on the Collins testbed.
When are data collected? It is estimated that data collection will take 1-2 days
Instruments/tools: Testing targets Edge components - flight oracle, stream receiver, upscaling, frame cache, streaming and Cloud components - image generator, virtual frame buffer, transcoder, media server. It also include common infrastructure representative of the CHARITY platform - Prometheus, Kubernetes, Grafana, Alert Manager
Methodology/Procedure: Pre-recorded flight data - gathered from real users interacting with in-house flight simulator - used for testing performance and scalability. Datasets are streamed to the flight oracle the same way data would be streamed from local users controls through the physics engine. From there, requests are routed to the Cloud Pod and resulting imagery streamed back to the Edge. For analysis, Cloud and Edge pods are co-deployed on a single node in the Collins infrastructure with the ability to add delays and jitter between them to simulate remote deployment.
Metrics to analyze the results • Trajectory Prediction accuracy • Frame rate received at client • Resolution received at client • Rendering feature enablement and disablement • Frame caching & cache retrieval latencies • Resource consumption - gpu/cpu, memory, network

3.12.2 Tests, data collection and analysis

The Flight Simulator Trainer Use Case necessitated wholescale design and development from scratch for a large number of components to move from the conventional monolithic deployment model to a distributed cloud native model that leverages AI services at the edge to offer latency optimizations for the cloud. Much of the experimentation and validation work so far has been focused on getting an operational model, achieving production level configurability to explore software adaptivity in conjunction with Task 3.3 and exploring the feasibility of using AI services at key points in the pipeline.

Unless stated otherwise, results presented have been gathered on the Collins testbed - the core of which is equipped with an Intel i9 processor and NVIDIA RTX 3090 GPU.

3.12.2.1 Test 1: Performance of AI Resolution Upscaling

In order to generate low resolution imagery on the cloud, we must be able to upscale it in real-time at the edge. We experimented with a number of deep learning tools for upscaling and focused on tools that did not require custom training as we felt these offered the most flexibility and wider applicability. The approaches evaluated were:

- Bicubic Interpolation: very fast (.007 seconds per frame but blurry images)
- EDSR (Enhanced Deep Super Resolution)
- ESPCN (Efficient Sub-Pixel Convolutional Neural Network)
- FSRCNN (Fast Super-Resolution Convolutional Neural Network)
- LAPSRN (Laplacian Pyramid Super-Resolution Network)
- SRGAN (Super-Resolution Generative Adversarial Network)
- ESRGAN (Enhanced Super-Resolution Generative Adversarial Network)

Below in *Table 21* we see the results we gathered while evaluating different approaches. In cases where the performance or quality was too poor, we discontinued and advanced onto the next alternative.

Table 21. Experimental results of evaluating upscaling techniques

Hardware	Method	Execution Time	Input Image	Upscale Resolution	FPS	VAMF Score[7]	CPU Data Transfer
NVIDIA GeForce RTX 3090	FSRCNN	0.01 sec	640*480	2560*1920		22	
	EDSR	2.27 sec	640*480	2560*1920		12	
	LapSRN	0.007 sec	640*480	2560*1920		20	
	ESPCN	0.93 sec	640*480	2560*1920		24	
	SRGAN	0.33 sec	640*480	2560*1920	3		
	ESRGAN	0.05 sec	320*240	1280*960	80	70	200,704 bytes
		0.09 sec	480*360	1920*1440	40	85	488,621 bytes
		0.15 sec	640*480	2560*1920	24	89	757,760 bytes
NVIDIA RTX A4000	ESRGAN	0.11 sec	320*240	1280*960	40	70	200,704 bytes
		0.23 sec	480*360	1920*1440	16	85	488,621 bytes
		0.31 sec	640*480	2560*1920	12	89	757,760 bytes

During testing of these approaches, we identified a debilitating bottleneck when attempting to transfer upscaled images from the GPU to the host. Upscaling on the GPU itself was very fast but getting access to the upscaled image so we could stream it or cache it was orders of magnitude slower. We tried a wide range of tactics to reduce this cost. A significant challenge with the frontrunner approach (ESRGAN) is the GPU memory consumption. We were observing consumption exceeding 20GB which made the approach untenable. With tuning, we found we could pin the memory consumption to approximately 9GB which was still very high but workable. We tried various techniques to improve the GPU transfer time and cost:

[7] VMAF is a perceptual video quality assessment algorithm developed by Netflix. It is designed to estimate the quality of videos as perceived by human viewers. We used it during our experiments to evaluate the quality of the produced imagery.

- Batching: Upon experimenting with submitting batches of frames for upscaling, we quickly exhausted available memory so had to abandon this approach.
- Multiprocessing: We tried using queues and multiprocessing on the CPU and again ran out of memory.
- GPU Arrays: We tried using GPU arrays (cupy) but this did not prove fruitful. Similar results to batching.
- We tried compression on the GPU before transferring to the CPU but the three-dimensional tensor output from the ESRGAN approach was not amenable to this.

The only approach to reduce the execution time was to reduce the amount of data we needed to transfer from the GPU, and this entailed reducing the resolution we could achieve with upscaling.

Currently, the approach is only feasible for upscaling from input images of 320x240 to 1280x960px. It demonstrates the concept of real-time resolution upscaling to a high quality but falls far short of the kind of performance we require. We have plans to evaluate an application-specific approach (LAPSRN, FRSCNN) to train a neural net on 4K video from FlightGear and evaluate the performance of the resulting model as other researchers have proposed that such a model delivers good quality results even on a CPU.

3.12.2.2 Test 2: Performance of AI Frame Rate Upscaling

Although modern XR headsets come equipped with FPS upscaling, we set out to investigate doing frame rate upscaling on the Edge. This offers a means independent of headset choice to ease experimentation while additionally enabling us to investigate the latest developments in frame interpolation that may not have made their way into commercial headsets.

We investigated two approaches:

- Lucas-Kanade Optical Flow: estimates the motion vectors (displacements) of image features between two frames and uses these to guide the generation of new frames
- RIFE (Real-Time Intermediate Flow Estimation): Deep learning technique using Convolutional Neural Networks. Estimates optical flow in both directions. Also seeks preserve temporal consistency and is designed to work in real-time.

From the beginning RIFE produced clearly superior results and we observed the ability to upscale from 10FPS to 40FPS across a range of resolutions with sub-second performance.

3.12.2.3 Test 3: Performance of AI Trajectory Prediction

We adopted an LSTM approach for trajectory prediction and trained a model using a small number of recorded flight trajectories and tested with an unseen trajectory. The position of an aircraft is captured by a set of values for latitude, longitude, heading, altitude, pitch and roll. Of these figures, we would expect a fast-moving commercial aircraft to experience most change on the geographical coordinates – latitude and longitude – and this has been borne out with our predictions which demonstrate prediction errors on these vectors. Our results can be viewed below in *Figure 16*.

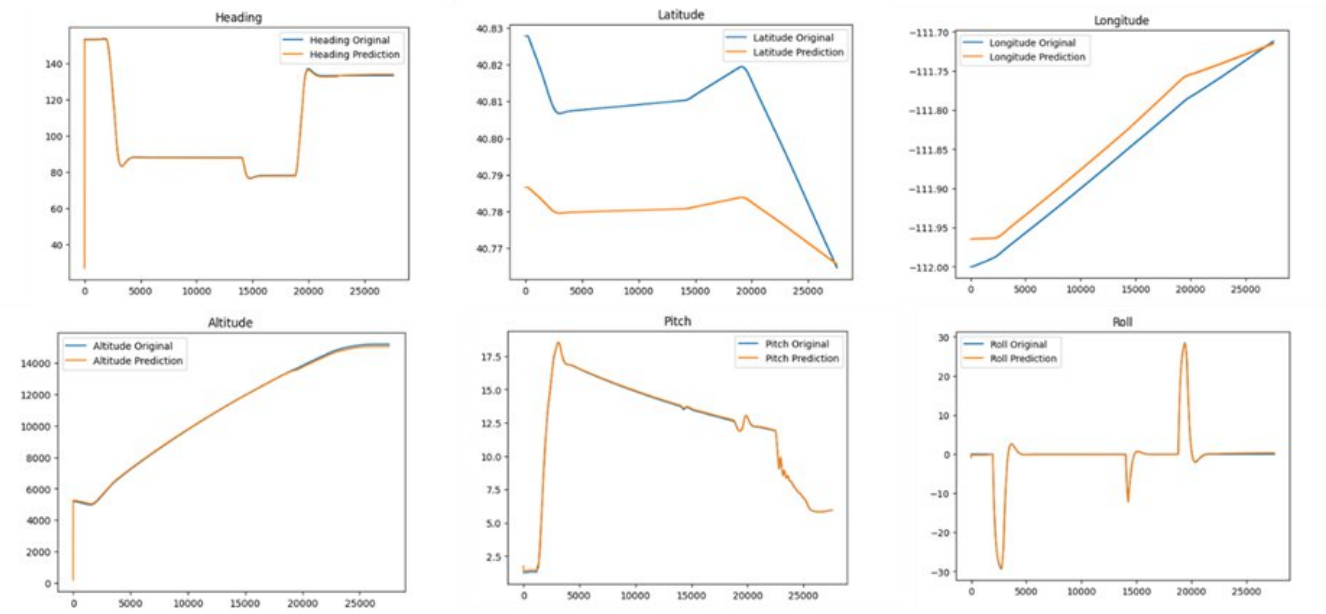

Figure 16.Predicted trajectory versus observed trajectory

While we observed deviations between the predicted positions and observed positions, we believe we have achieved sufficient accuracy to demonstrate the prediction concept.

3.12.2.4 Test 4: Software Adaptation

Adaptivity was a key target for the Flight Simulator Use Case from the outset. We sought to facilitate configurable rendering sophistication such that we could alter the weather effects and enable or disable advanced rendering features such as shadows and reflections. Additionally, we sought to support configurable frame rate and resolution. To facilitate integration with the Dynamic Software Adaptation model in Task 3.3, the use case needed to support a coherent and joined-up configurability model that would orchestrate configuration per user across multiple services[8] even when we do not have access to the source code of those services.

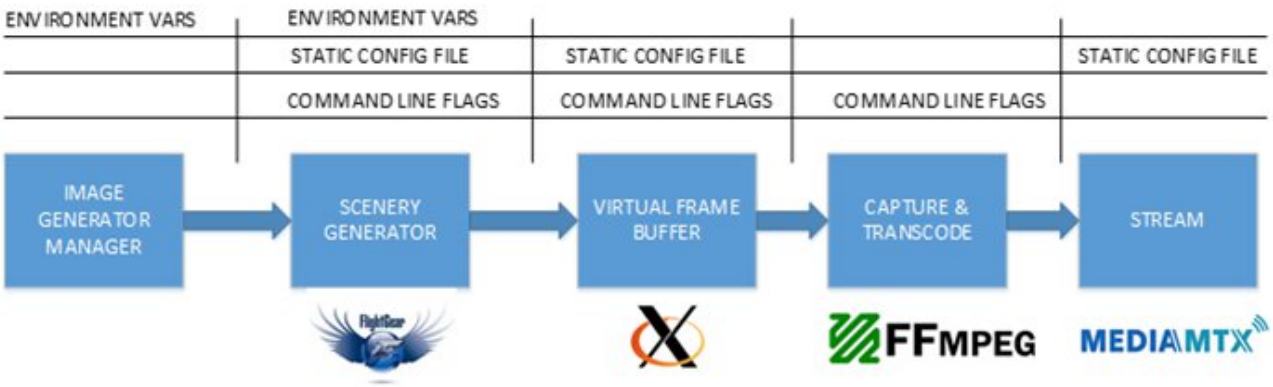

Figure 17. Diversity of configuration channels for remote rendering components

[8] To support a configurable frame rate, for example, requires the FlightGear Image Generator to be instructed to operate at a particular frame rate generation cadence; for this frame rate to be supported by the virtual frame buffer; for the ffmpeg transcoder to acquire frames at this rate and stream them to the RTSP server at this rate; and for the reader of the resulting video stream exposed by the RTSP server to be aware of the incoming frame rate target to operate correctly. We cannot require this to involve multiple configuration settings in different services as this would be too error prone.

After containerizing all services with Docker, we first implemented a co-ordinated configuration scheme using Docker Compose to centralize configuration for groups of containers and then evolved this to use Kubernetes ConfigMaps. We deployed Prometheus, custom exporters, and its Alert Manager to facilitate the configuration of environmental triggers (such as GPU busy-ness) that could initiate a Kubernetes rolling update – the launching of alternatively configured pods to seamlessly take over the operation of existing pods and in effect offer dynamically adapted software without session interruption.

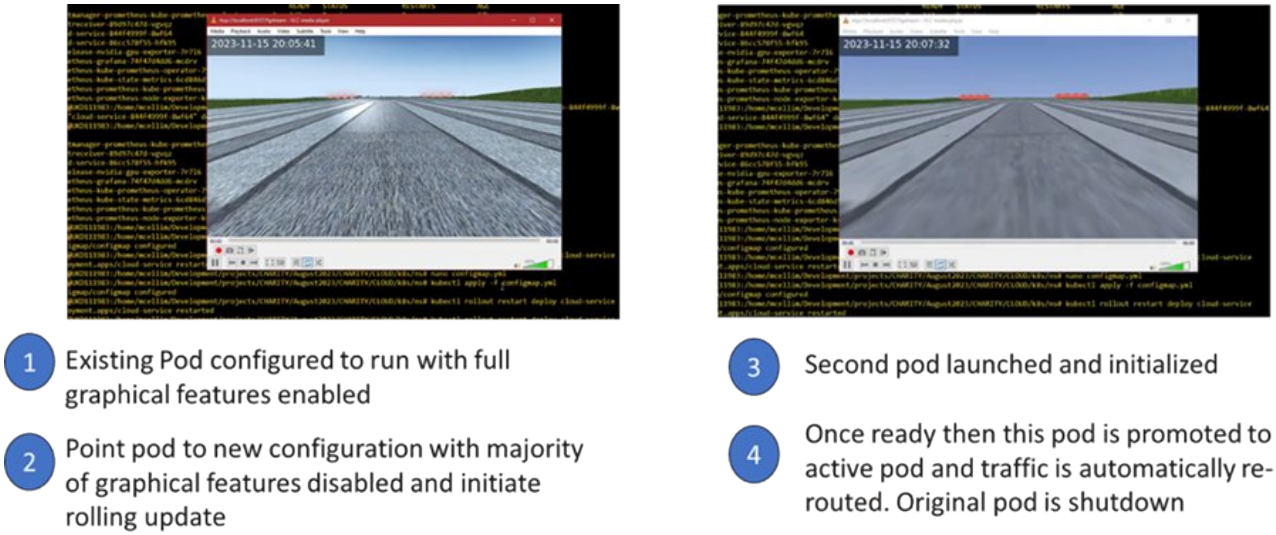

Figure 18. Dynamic Software Adaptation using rolling updates for the Collins use case

Although we were able to validate the model experimentally, there are shortcomings that we plan to address. Primarily this is due to the warm-up delay for a newly launched instance of the FlightGear service. Between populating internal caches, displaying splash screens and cycling through integrity checks the service can take tens of seconds to launch. This is resulting in a pronounced stutter with our handover which we plan to address using readiness probes such that traffic is not handed over from the currently operating configuration until the new configuration has fully started up.

We ran various experiments to observe the variations in resource consumption of the flight simulator Cloud Pods under different configurations and results are shown below in *Table 22*. The flight simulator can be run with just a single window (showing the scenery straight ahead) or multiple windows for left and right views[9]. Low QModes signify operation with disabled advanced graphical features (smoke, shadows, etc.) while high QModes signify operations with all features enabled.

The bandwidth reflects the amount of data being sent from the transcoder to the streamer.

Table 22. Resource usage profiles across various configurations

QMode	GPU Memory usage (MiB)	GPU utilization	Frames Per Second (FPS)	Resolution	Bandwidth (MB/sec)
Low single window	105	2%	10	848x480	0.1
Low single window	120	4%	20	848x480	0.16
Low single window	122	13%	60	848x480	0.38
High single window	208	3%	10	848x480	0.51

[9] We encountered a persistent problem with the display of the right-hand window with FlightGear that we have not yet succeeded in solving. The results with multiple windows only represent two windows.

High single window	208	6%	20	848x480	0.75
High single window	208	18%	60	848x480	1.2
Low multiple window	270	8%	10	848x640	0.3
Low multiple window	284	16%	20	848x640	0.35
Low multiple window	316	37%	60	848x640	0.5
High multiple windows	675	13%	10	848x640	1.35
High multiple windows	675	29%	20	848x640	1.75
High multiple windows	675	33%	60	848x640	2.4
High single window	268	6%	10	1920x1080	2.1
High single window	268	16%	20	1920x1080	2.6
High single window	268	19%	60	1920x1080	3.2

The experiments revealed the somewhat surprising effects of advanced graphical flourishes on the bandwidth requirements for video streams. We witnessed a five-fold increase in bandwidth between a stream with advanced graphical features turned on (High Single Window) versus the same stream with the features turned off (Low Single Window). The relationship between bandwidth and graphical effects can be explained by the amount of additional variance that graphical effects (such as rain, shadows, smoke, etc.) produce across video frames and thus increasing the amount of change from one frame to another – thus reducing the benefits of savings that can be achieved with video codecs.

Graphical flourishes are not easily recovered if not included at rendering time so while we can indeed reduce the resource footprint, we cannot do so without clearly visible degradation of service to the end user. Resolution and frame rate are aspects we can seek to degrade at the rendering source with the objective to recover them closer to the user. From the experimental results shown in Table xx we can observe the effect of going from Standard Definition (848x640) to High definition (1920x1080) resolution at 10 frames per second is an effective quadrupling of the bandwidth needs and doubling of the GPU utilization.

3.12.2.5 Test 5: Assessing Edge Costs for Cloud Savings

Operating with reduced cloud resources means introducing additional compensating resources near the user if we want to try and maintain a similar Quality of Experience. If we operate at reduced frame rate and resolution at the remote rendering source on the cloud, then we sought to quantify what the cost of recovering this loss of fidelity could be. We conducted an experiment targeting 60 frames per second of Full High-Definition resolution. This required approximately 20% of our GPU, 1% of GPU RAM and 2.6 MB/s. All advanced graphical features were turned on. This is summarised below in *Figure 19*.

Figure 19. Generate high quality at rendering source

Reducing the frame rate at source to 15 and the resolution to Standard Definition reduced our bandwidth needs from the cloud by approximately 90% and the GPU consumed on the cloud by 75%. Clearly significant savings. However, attempts to recover this loss at the edge requires the use of additional physical resources. For the purposes of our experiment, the edge node is the same machine as used for the cloud, so this gives us a like-for-like comparison. The resolution upscaling model we are using is very demanding on GPU RAM so we see very significant uptick in this area. The results are summarized below in *Figure 20*.

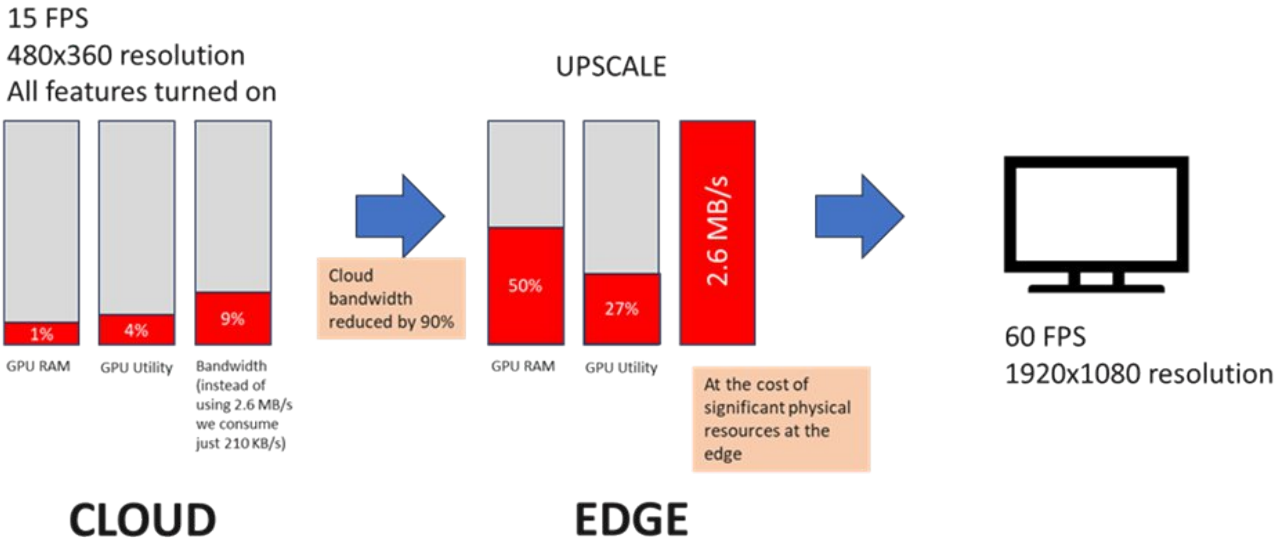

Figure 20. Generate low quality on the cloud and seek to recover quality at the edge. Significant bandwidth reductions but also significantly increased resource usage overall

We mentioned earlier when discussing the tests carried out on our resolution upscaling, we plan to experiment with a more lightweight model trained specifically for the our application.

3.12.2.6 Test 6: Turnaround performance

We have encountered numerous obstacles with using third party opensource components such as FlightGear, XDummy, FFMPEG and MediaMTX as a result of documentation shortcomings and lack of technical support – particularly FlightGear. This has resulted in significant delays and difficulties assembling a performant pipeline and continues to cause challenges. The behaviour and tuning of a ring buffer within the MediaMTX server, for example, is very sensitive to the frame rate and resolution

being streamed and even small imbalances manifest in significant delays and dropped frames - yet its operation remains opaque, undocumented, and buried deep within the codebase.

Ultimately, media is streamed from an in-memory cache at the edge using predicted positional data which reduces the uncertainty of shuttling data over Internet links. However, cache speeds have limits and moving large video frames takes time. Experiments with Redis – a leading in memory cache – showed that writing a 4K video frame to cache takes 0.015 seconds (consuming around 25MB of RAM) and reading such a frame from cache takes almost twice as long at 0.027 seconds. If we do nothing else but read such frames from a cache, we can operate no faster than 35 frames per second on the testbed. For comparison, a 1K video frame takes just 0.002 seconds to write and 0.006 to read meaning we could operate at around 175 frames per second. These results demonstrated that it is not feasible to achieve 4K resolution at 60 FPS cached and streamed from the edge. 4K resolution is only feasible at low frame rates which may still be ok if the frame rate upscaling can occur in the XR headset but further experimentation with the use case is now targeting 1K resolution[10] as we prioritize validating the concept over dealing with the hardware constraints at our disposal.

3.12.2.7 Test 7: Scalability

We have designed and developed the use case to deploy two Kubernetes PODs per user – one on the cloud and one on the edge. Apart from a shared monitoring infrastructure, there are practically no shared components between users. This model enables us to cleanly adapt the Quality of Experience per user according to their location, priority, and device characteristics. It also makes scaling very predictable. If a single user requires resources X then two users will require resources 2X and so on.

3.12.3 KPIs assessment and requirement satisfaction

The current assessment of how the Flight Simulator Use Case satisfies the original requirements identified is presented below in *Table 23*.

Table 23. Requirements status for Flight Simulator Use Case

Requirement	Description	Status
		Implemented
		Not yet but will be
		Issues which may not be overcome
F_UC3_13	The simulation must facilitate collaboration between users to efficiently execute the simulated mission	On course. Functionality in place to enable users to join shared arena and view each other's aircraft when in proximity
F_UC3_14	Scenery generation may support scenery with different weather	Implemented and validated
F_UC3_15	The simulated environment should allow participants to join or leave simulation at any time	On course – see F_UC3_13
F_UC3_16	The simulation should enable prediction of background scenery demands so that it can be	Implemented and validated

[10] It is important to point out that the flight simulator has multiple windows – not just one. If we want 4K high resolution frames across all windows then the resources required would be beyond what we currently have available.

	pre-fetched by any component from off-line storage	
F_UC3_17	The simulation should enable custom tiling of cloud-based image generator output to facilitate variable resolution across a single frame	Validated and implemented. Scenery segmented into side and front windows with separate streams
NF_UC3_18	The simulation should adapt imagery frame rate and resolution in accordance with available bandwidth, observed latency, and user equipment capabilities	Validated and implemented.
NF_UC3_20	The RTT from user action to presentation of updated imagery should be < 15ms	Still under investigation
NF_UC3_21	Number of concurrent users (virtual & real) in a single simulation scenario should be > 30	Still under experimentation. We plan to demonstrate small number of users within the confines of available hardware resources and demonstrate how further scalability can be extrapolated
F_UC3_22	The simulation should be able support both active participants (present in the simulated environment) and passive observers (not present in the simulate environment)	Implemented and validated. We can attach multiple viewers to a given RTP stream
NF_UC3_23	The video resolution of presented imagery must be greater than 60 FPS 4K.	Under investigation. Highly dependent on headset motion smoothing capabilities
NF_UC3_11	The simulated environment must provide a consistent simulation state across all users, including rendering of other user activities	On course – see F_UC3_13

In terms of KPIs, the following were listed at the outset of the project.

- KPI-UC-3.2 RTT (aeronautical) – sum of network latency and game server response time < 15ms
- KPI-UC-3.3 Number of CCUs>30
- KPI-UC-3.5 Data services required (raw data streaming, rendering, compression, caching, encoding) >=5

The first two have already been discussed previously while the third regarding data services has been comfortably exceeded.

4 Lessons learnt

In this section the main outcomes from the evaluation subtopics and the assessment of the KPIs during this phase are summarized in an effort to guide the final evaluation phase.

The testing phase of integrating **High level orchestrator** algorithms and solutions with the platform is ongoing. While the solution's effectiveness has been proven through simulations and on simple platforms, further data collection is underway to assess its performance with more complex scenarios. This includes analysing its support for an increasing number of virtual clusters and a wider range of deployed applications, encompassing both example blueprints and those from project use cases.

The **Low-Level Orchestrator** functional capabilities were demonstrated in the EUCNC & 6G Summit 2023, during the booth exhibition through live demonstrations. A preliminary Prometheus-based deployment was used to monitor, collect, and show metrics about the LLO clusters, applications and links operations exploiting the testbed resources of CloudSigma. In summary, these preliminary tests underscore the capabilities of the Low Level Orchestrator in dynamically managing Kubernetes clusters and deploying containerized applications, as well as its effectiveness in facilitating cross-cluster networking and supporting distributed services in multi-domain edge environments.

The **Monitoring framework** tests provide valuable insights into the performance and responsiveness of CHARITY's architecture components. Through rigorous evaluation of Resource Indexing, Monitoring Manager, and Monitoring Agents, several key findings emerge: a) Resource Indexing demonstrates its capability to offer real-time updates on cluster performance and response latency, crucial for maintaining system efficiency and reliability, b) The Monitoring Manager effectively handles requests and exhibits acceptable latencies, ensuring timely access to critical monitoring data such as metrics history and active alarms/alerts, c) Monitoring Agents, represented by Prometheus servers, prove reliable in gathering and delivering accurate application performance data, essential for informed decision-making and troubleshooting. Furthermore, the test scenarios—Stable Monitoring, Migration, and New Cluster—highlight the framework's adaptability to different states of the architecture, ensuring continuous monitoring and scalability.

The experimental results for the **Forecasting Model** compare a proactive horizontal scaling approach with a conventional reactive method across latency-related metrics, demonstrating the superiority of the proactive approach. Similarly, the comparison between an intelligent proactive approach (IPFT) and a conventional reactive method (RFT) across fault tolerance-related metrics reveals the proactive approach's dominance. These findings hold true even when assessing different task scheduling algorithms like Round-Robin, MinMin, and MaxMin. In summary, the proactive strategies consistently outperform reactive methods across all examined metrics, indicating their efficacy in enhancing system performance and fault tolerance.

The **Point Cloud encoder/decoder** component has undergone testing and integration into the UC1-3 pipeline (the CPU version). The component operates at an overall frame rate of approximately 5 fps, processing and streaming images at a resolution of 1280 x 752, 8 viewpoints simultaneously. This number of views allows users of the holographic display to make slight viewpoint adjustments without additional data transmission. Preliminary tests of the GPU version in isolation show promising results, with anticipated frame rates of over 30-40 fps, even on more complex scenes. In terms of KPIs assessment, the component meets KPI-4.3, which concerns specialized data services support, including streaming, rendering, compression, caching, and encoding. The potential performance of the GPU version satisfies the speed requirements necessary for the Holographic Assistant application to ensure a good Quality of Experience (QoE).

The **Mesh Merger** service is being integrated into the UC3-1 Collaborative Gaming Application. Tests conducted demonstrate sufficiently fast processing times, ensuring a good Quality of Experience (QoE) for gamers. Specifically, less than 2 seconds are required to download and process a new acquisition into the Mesh Collider. Efficient transmission is facilitated by employing a binary version of a JSON containing a PLY format of the mesh. Despite the format not being compact, the number of triangles per mesh collider is manageable and suitable for an interactive experience. In terms of KPI assessment,

it fulfills KPI-4.3, which pertains to specialized data service support, including streaming, rendering, compression, caching, and encoding.

For **UC1-1 and UC1-2** initial tests were conducted for video streaming over a wired local network aimed. With a 1GB wired connection, latency ranged from 5000-7000ms due mainly to local video manipulation component performance dependency. However, results were hardware configuration-dependent, with significantly higher latency than expected, prompting the need for a cloud-based solution with enhanced computing power. Transitioning to video streaming via a cloud server, initial attempts with various local and web streaming protocols yielded unsatisfactory results due to latency issues. After adopting the WebRTC protocol and conducting two 2-hour sessions, promising results were observed. Latency reduced to under 1000ms, and audio-video synchronization was achieved. The latency graph showed a relatively stable average latency of 150-200 ms, still deemed insufficient for the Holographic Concert and Holographic meeting se cases, pending completion and testing of the cloud video manipulation component. In terms of KPI assessment, KPI-UC1-2.1 aimed for an average latency of <20ms, which was not met. Further testing post-completion of video manipulation and synchronization components is necessary. However, KPI-UC-1-2:3, which pertains to required data services, was satisfied with tests conducted involving transcoding, rendering, and networking services.

For **UC1-3 (Holographic Assistant)** the tests evaluated the overall ecosystem's performance and operation, revolving around a specific use scenario. The latency between providing eye-coordinates and rendering new views from the 3D point cloud consistently remains below 60 ms. The frame-rate of the streamed 3D point cloud currently stands at approximately 5 FPS, but with GPU optimizations, this can be increased to 30 FPS or more. The delay between sending and receiving 3D point cloud data is around 3-4 seconds, primarily due to compression and buffering processes. In terms of KPI assessment KPI-UC-1.1 ensures that the average latency between sending input data and receiving 3D point cloud data is ≤60ms, which is consistently achieved. KPI-UC-1.5 focuses on the latency in speech input and output, aiming for ≤2 seconds. Typically, reaction times are below 2 seconds, but may vary depending on the load on speech recognition services. KPI-UC-1.3 targets a holographic visualization frame rate of ≥30Hz. While the computation load is typically below 50%, resulting in a frame rate above 30Hz for holographic visualization, optimizations are required to increase the frame rate of the content from edge to client, currently limited to 5 Hz. Further optimizations, especially utilizing GPU processing, are expected to achieve the desired frame rate easily.

For **UC2-1 (Medical Training)** several testing sessions were towards reaching a target of 50 Concurrent Connected Users (CCUs) by the project end. These tests involved real users with Head-Mounted Displays (HMDs) and simulated users via bots. The experiments aimed to assess Use Case components separately, focusing on deployment in testbeds provided by the project and evaluating metrics and Key Performance Indicators (KPIs). HMDs were provided by ORAMA, and participants from the ORAMA team in Greece were involved. The LSPart1 and LSPart2 components were deployed in the CloudSigma testbed in Sweden. Tests were conducted with a varying number of VMs with GPU acceleration for HMD users and simulated users, in total 24 users. Metrics such as latency, frames per second, packet loss percentage, and sent rate were recorded. In terms of KPI assessment KPI-UC2.1 aimed for an average latency of <20 ms, which was achieved as encoding/decoding latency remained below 20 ms. KPI-UC-2.2 aimed for more than 50 CCUs, which was partly reached with 24 users (4 HMDs and 20 bots). KPI-UC-2.3 aimed for more than 5 different VR HMDs, with tests conducted using 4 HMDs and plans to increase the variety in future experiments. KPI-UC-2.4 focused on required data services, with experiments covering rendering, physics, networking, and communication services. KPI-UC-2.5 targeted automated configurable soft-body simulation for objects with a large number of vertices, which was achieved with a medical sample app from ORAMA featuring objects with over 8000 vertices, enabling configurable soft-body simulation.

For the **UC2-2 use case (VR Tour Creator)** several experiments and tests were conducted to develop a functional prototype of a real-time 360 video experience. The use of the WebRTC protocol proved successful after unsuccessful attempts with HLS, LHLS, and DASH due to excessive delay. Livestreaming VR functionality was implemented with a metrics middleware to evaluate bandwidth and video condition parameters. A 5.7k video was streamed to the cyango-media-server and consumed via the

cyango-story component across different premises during a 2-hour session, yielding various relevant metrics. The video converting case made significant progress in testing and implementation. It involves continuous improvement and trials with multiple video formats and sizes to enable users to upload and stream videos effectively. Limitations on video quality for VR headsets, particularly concerning bitrate and resolution, were identified. Tests with various video formats, resolutions, and bitrates using different ffmpeg commands led to the creation of a streamable HLS playlist for videos up to 500 Mb, although further improvements are required. An analytics architecture was also implemented to store metrics about file converting performance and VR video streaming latency. Regarding KPI assessment KPI-UC2.1 aimed for an average latency of <20 ms, achievable under optimal conditions despite external factors. KPI-UC-2.2 targeted more than 50 concurrent users (CCUs), but testing is limited by cloud resources, with only 2 concurrent users streaming/transcoding video currently supported. KPI-UC-2.3 aimed for more than 5 different VR Head-Mounted Displays (HMDs), with streaming tested on 1 VR HMD and plans to test with 5 concurrent users soon. KPI-UC-2.4 focused on required data services, including transcoding VR video, rendering VR video and 3D, and networking, all successfully tested.

For **UC3-1 (Collaborative Gaming)** the ongoing development and testing of gaming components within the CHARITY platform involve various elements, including Game Clients (iOS app), Game Server (Docker image), Mesh Merger (Docker image), and Game Servers Managers. The Game Server Docker image has been configured and prepared for manual deployment within the project's infrastructure, enabling successful testing of connections between Game Clients and Game Servers. Work is underway to develop and prepare the Game Servers Manager for fully automatic deployment of Game Servers. Testing focused on CHARITY AMF API connectivity and functions, with plans for further tests once orchestration components are operational. Latency measurement tools are built into Game Servers Manager, Game Server, and Game Clients, covering response time computation during game runtime. In terms of KPI assessment: KPI-UC-3.1 targets a Round-Trip Time (RTT) for gaming, with the sum of network latency and game server response time aimed to be <100ms. Progress toward this KPI has been substantial, with the current latency well within the target, indicating a highly responsive gaming environment.

The **UC3-2** underwent comprehensive design and development efforts, transitioning from a conventional monolithic deployment model to a distributed cloud-native model. Much of the experimentation focused on achieving operational readiness, configurable adaptivity, and leveraging AI services at the edge for latency optimizations. Seven tests were primarily conducted on the Collins testbed, equipped with an Intel i9 processor and NVIDIA RTX 3090 GPU. In terms of KPI assessment, for the RTT the sum of network latency and game server response time, the target of <15ms looks attainable but has not yet been demonstrated as there is some outstanding development work. For the Number of CCUs the requirement with over 30 Concurrent Connected Users (CCUs), while not demonstrated, looks on course to be achievable. For the Data services it exceeds the requirement of >=5 data services required, demonstrating ample coverage in this aspect. Overall, the Flight Simulator Trainer Use Case has made significant progress in meeting its original requirements and KPIs, with ongoing efforts focused on addressing remaining challenges and optimizing system performance.

5 Conclusions

The document presents the outcomes of the first version of the validation and evaluation of components and services of the CHARITY prototype as well as the Use Cases in different testbeds environments. The experimentation involves the evaluation of individual subtopics which address core functionalities and aspects of the platform and are linked to the functional and non-functional requirements. The evaluation of each subtopic was undertaken by the partner responsible for developing the respective modules of the platform, services and use cases, the procedure and metrics were detailed upon which the analysis was conducted and reported in the present deliverable. Furthermore, based on the related KPIs the analysis comprised of different experiments. The outcomes of the evaluation are summarized as lessons' learnt. In the upcoming final version of the deliverable all Use Cases, with the exception of the Use Case UC1-3 Holographic Assistant (due to SRT leaving the consortium) will be evaluated exploiting the deployment, monitoring and orchestration features of the CHARITY platform.

[end of document]