



D4.1 SPIRIT PLATFORM (FIRST VERSION)

Revision: v.1.1

Work package	WP 4
Task	Task 4.1, 4.2, 4.3, and 4.4
Due date	30/09/2023
Submission date	12/05/2024
Deliverable lead	Deutsche Telekom
Version	1.1
Authors	Vivien Helmut - Editor (DT), Jeroen van der Hoof, José Santos (IMEC), Nick Turay (EDD), Christoph Stielow (TSI), Peter Hofmann (DT-Sec), Sergio Tejada Pastor (Fraunhofer), Hermann Hellwagner, Minh Nguyen, Shivi Vats (UNI-KLU), Peng Qian (SURREY)
Reviewers	Tim Wauters (IMEC)
Abstract	This report documents the integration activities of the project, with a focus on the actual software development of the first version of the SPIRIT system platform for supporting heterogeneous telepresence use case applications and components. This process is driven by the development and integration of a set of project-designated use cases, system validation and testing.
Keywords	Development, telepresence, holographic communication, testing, evaluation.

www.spirit-project.eu



Grant Agreement No.: 101070672
Call: HORIZON-CL4-2021-HUMAN-01

Topic: HORIZON-CL4-2021-HUMAN-01-25
Type of action: HORIZON-RIA

Document Revision History

Version	Date	Description of change	List of contributor(s)
V1.0	31/12/2023	First published version	Jeroen van der Hooft, José Santos (IMEC), Nick Turay (EDD), Vivien Helmut (DT), Christoph Stielow (TSI), Peter Hofmann (DT-Sec), Sergio Tejeda Pastor (Fraunhofer), Hermann Hellwagner, Minh Nguyen, Shivi Vats (UNI-KLU), Peng Qian (SURREY)
V1.1	10/05/2024	Review Comment Revision	

DISCLAIMER

The information, documentation and figures available in this deliverable are written by the "Scalable Platform for Innovations on Real-time Immersive Telepresence" (SPIRIT) project's consortium under EC grant agreement 101070672 and do not necessarily reflect the views of the European Commission.

The European Commission is not liable for any use that may be made of the information contained herein.

COPYRIGHT NOTICE

© 2022 - 2025 SPIRIT Consortium

Project co-funded by the European Commission in the Horizon Europe Programme		
Nature of the deliverable:	to specify R, DEM, DEC, DATA, DMP, ETHICS, SECURITY, OTHER*	
Dissemination Level		
PU	<i>Public, fully open, e.g. web (Deliverables flagged as public will be automatically published in CORDIS project's page)</i>	✓
SEN	<i>Sensitive, limited under the conditions of the Grant Agreement</i>	
Classified R-UE/ EU-R	<i>EU RESTRICTED under the Commission Decision No2015/ 444</i>	
Classified C-UE/ EU-C	<i>EU CONFIDENTIAL under the Commission Decision No2015/ 444</i>	
Classified S-UE/ EU-S	<i>EU SECRET under the Commission Decision No2015/ 444</i>	

* R: Document, report (excluding the periodic and final reports)

DEM: Demonstrator, pilot, prototype, plan designs

DEC: Websites, patents filing, press & media actions, videos, etc.

DATA: Data sets, microdata, etc.



DMP: Data management plan

ETHICS: Deliverables related to ethics issues.

SECURITY: Deliverables related to security issues

OTHER: Software, technical diagram, algorithms, models, etc.



EXECUTIVE SUMMARY

Telepresence can be seen as the next generation of communication applications that will significantly enrich the human-to-human and human-to-machine experience, blurring the boundaries between the physical and virtual worlds. Until now, telepresence solutions have been high-end, expensive, and quite conventional audio and video conferencing systems. However, the last decade has seen intensive research and development on VR/AR/XR technologies and applications that have significantly improved the state of the art.

The goal of the SPIRIT project is to realise Europe's first multi-site, interconnected framework dedicated for supporting the operation of heterogeneous collaborative telepresence applications at large scale through relevant technology innovations.

This report is a SPIRIT deliverable of the work package 4 “Platform Development, Integration and Validation”. It presents the first version of the SPIRIT Platform. The work mainly focuses on the actual software development of the SPIRIT system platform for supporting heterogeneous telepresence use case applications and also the integration of the platform with the distributed, interconnected network infrastructures underneath. This process is driven by the development and integration of a set of project-designated use cases.

This report

- provides an overview of the 5G testbeds in Surrey and Berlin, e.g. the available 5G network infrastructure and the computing power of the Edge Cloud located there.
- lists partner components that were integrated into the testbeds during the project and documents the findings of this integration process.
- covers the implementation of partner use cases, integration of the platform components, and the description of the necessary adjustments for a smooth integration into the testbeds, as well as a look ahead to how these components might find use in various scenarios, such as third-party applications.
- specifies an approach for deployment of Confidential Computing-protected VMs to ensure data owner’s control of personal data.
- presents the subjective Quality of Experience (QoE) tests conducted by partners to assess the impact of quality, quality switching, viewing distance, and content characteristics on the perception of point clouds in AR environments.

The initial work consisted of integrating the partner components into a common testbed environment. In the next step, the two testbeds are connected so that the use cases, application frameworks can be deployed and validated across the testbeds.

In parallel, an Open Call will be launched inviting third parties to realize their telepresence use cases experimentally. The platforms and components described in this document are available for this purpose and can be integrated into their use cases. The results from the experiments will be used by the project to improve the existing partner components and to develop new innovative components or connections.

There will be two more iterations of this document. In the next iteration, further enhancements and improvements added to the first version of the SPIRIT platform will be documented in a second version and at the end of the project a report about the final version of the SPIRIT platform is planned.

TABLE OF CONTENTS

Disclaimer	2
Copyright notice	2
EXECUTIVE SUMMARY.....	4
LIST OF FIGURES.....	8
LIST OF TABLES	10
ABBREVIATIONS.....	11
1 INTRODUCTION	15
1.1 Purpose of the Document	15
1.2 Structure of the Document	16
2 TESTBEDS	17
2.1 Surrey.....	17
2.1.1 <i>General Information on University of Surrey Testbed</i>	17
2.1.2 <i>Testbed Infrastructure</i>	18
2.1.3 <i>Onsite Use Cases</i>	19
2.1.4 <i>Onsite Collaborative Opportunities</i>	19
2.2 Berlin	19
2.2.1 <i>General Information Deutsche Telekom Testbed</i>	20
2.2.2 <i>Testbed Infrastructure</i>	20
2.2.3 <i>Onsite Use Cases</i>	21
2.2.4 <i>Onsite Collaborative Opportunities</i>	21
3 PLATFORM COMPONENTS	23
3.1 Live Multi-Source Holographic Streaming	23
3.1.1 <i>Deployment of the Frame Render and Synchronization Function</i>	23
3.1.2 <i>Validation</i>	23
3.2 Network-aware Resource Scheduler (Diktyo)	24
3.2.1 <i>Integration with CloudNativeLab</i>	25
3.2.2 <i>Integration with Berlin Testbed</i>	27
3.3 Holographic Human-to-human Communication	27
3.3.1 <i>Common Prerequisites for Integration</i>	28
3.3.2 <i>Compatibility Test of Client Devices</i>	28
3.3.3 <i>Integration Efforts</i>	29
3.3.4 <i>Additional Components</i>	30
3.3.5 <i>Validation</i>	31
3.3.6 <i>Summary</i>	31
3.4 Real-Time Animation and Streaming of Realistic Avatars	32

3.4.1	<i>Common Prerequisites for Integration</i>	32
3.4.2	<i>Containerization of the Application</i>	32
3.4.3	<i>Deployment and Interconnection of Containers</i>	33
3.4.4	<i>Compatibility Test of Client Devices</i>	34
3.4.5	<i>Validation and Performance Evaluation</i>	34
3.4.6	<i>Summary</i>	34
4	IMPLEMENTED USE CASES	35
4.1	UC Hologram: Holographic Human-to-human Communications	35
4.1.1	<i>Overview</i>	35
4.1.2	<i>Extension of Use Case with Third Party Applications</i>	36
4.1.3	Summary Validation	39
4.2	UC Multi-source: Live teleportation with 5G MEC	39
4.2.1	<i>Overview</i>	39
4.2.2	<i>Use of the System within the Surrey Testbed</i>	39
4.2.3	Summary Validation	41
4.3	UC Avatar: Real-Time Animation and Streaming of Realistic Avatars	41
4.3.1	<i>Overview</i>	41
4.3.2	<i>Use of the System within the Berlin Testbed</i>	42
4.3.3	<i>Extension of Use Case with Third Party Applications</i>	43
4.3.4	Summary Validation	45
4.4	Real-Time Human-Machine Interactions	45
4.4.1	UC Robot: Distributed Steering of Autonomous Mobile Robots (AMRs).....	45
4.4.2	Summary Validation	47
5	SECURITY DEVELOPMENT AND TECHNICAL INTEGRATION	48
5.1	Problem Statement.....	48
5.2	Overview of this Section	49
5.3	Overview of Linux-based Confidential Computing on Bare Metal Server	50
5.3.1	<i>Trust Model</i>	50
5.3.2	<i>Basic Approach to Measured Start of VM</i>	51
5.3.3	<i>Measured Start with Fully Encrypted Disk Images</i>	52
5.3.4	<i>Application Example: Secure Certificate Provisioning</i>	53
5.4	Specification of Components	55
5.4.1	<i>Setup Phase</i>	55
5.4.2	<i>Operation Phase</i>	58
6	QUALITY OF EXPERIENCE EVALUATION	60
6.1	A Platform for Subjective Quality Assessment of Point Clouds in Augmented-Reality Environments	61
6.1.1	<i>Point Cloud Previews</i>	61
6.1.2	<i>Subjective Testing</i>	62

6.2	Dataset and Results of Subjective Tests for Point Clouds in Augmented-Reality Environments	63
6.2.1	<i>Experiment Tasks</i>	63
6.2.2	<i>Dataset Description</i>	66
6.2.3	<i>Subjective Test Results</i>	68
6.3	QoE Models	71
6.3.1	<i>Machine Learning-Based QoE Models</i>	71
6.3.2	<i>Fine-tuned P.1203 Mode 0 Model</i>	73
7	CONCLUSIONS	75
8	REFERENCES	76
	APPENDIX A: SECURITY	80
A.1	Establishing the VPN Connection with the Lab Infrastructure	80
A.2	Access NAS Device for Software Download and VM Upload	84
A.3	Using the Supplied VM OVA File	86
A.4	End-to-end Example	98
A.5	Formal Definition of Remote Management Web Service (YAML)	105
A.6	Updating the Template Files	108
A.7	Setup on the Host System for new Guest VMs	110
A.8	Configure the REST Server as systemd Service	111
A.9	API Description	113
	APPENDIX B: SUBJECTIVE TEST PLATFORM	118
B.1	Platform Installation	118
B.1.1	<i>System Requirements</i>	118
B.1.2	<i>Software Pre-requisites</i>	118
B.1.3	<i>Preparing the Data</i>	118
B.1.4	<i>Setting Up the Project</i>	118
B.1.5	<i>Point Clouds Loader</i>	119
B.2	Platform Usage	119
B.2.1	<i>Point Clouds Preview</i>	119
B.2.2	<i>Subjective Testing</i>	119

LIST OF FIGURES

FIGURE 1: 5G TESTBED IN UNIVERSITY OF SURREY	17
FIGURE 2: INFRASTRUCTURE AND CONNECTIVITY	18
FIGURE 3: HOLOGRAPHIC SERVER AND CLIENT CONNECTED VIA 5G NETWORK.....	19
FIGURE 4: LOCATION OF DEUTSCHE TELEKOM TESTBED	19
FIGURE 5: LOCATION OF TESTBED IN BERLIN	20
FIGURE 6: LAB FLOOR WERNER VON SIEMENS CENTER	20
FIGURE 7: NETWORK INFRASTRUCTURE	21
FIGURE 8: KUBERNETES MANAGEMENT PLATFORM.....	22
FIGURE 9: DEPLOYMENT OF FRAME PRODUCTION AND SYNCHRONIZATION FUNCTION ON SURREY'S PLATFORM.....	23
FIGURE 10: VALIDATION OF THE FRAME PRODUCTION AND SYNCHRONIZATION FUNCTION ON SURREY'S PLATFORM.....	24
FIGURE 11: DIKTYO SCHEDULER RUNNING PROPERLY	25
FIGURE 12: DIKTYO SCHEDULER CONFIGURATION PARAMETERS.....	25
FIGURE 13: CLOUDNATIVELAB MAIN DASHBOARD	26
FIGURE 14: CLOUDNATIVELAB K8S CLUSTER CREATION PAGE	26
FIGURE 15: DEPLOYMENT EXAMPLE THAT SPECIFIES SCHEDULER FOR POD DEPLOYMENT.....	26
FIGURE 16: DEPLOYED PLATFORM FOR HOLOGRAPHIC HUMAN-TO-HUMAN COMMUNICATIONS	27
FIGURE 17: CONNECTIVITY BETWEEN DEPTH CAMERA AND EDGE CLOUD WITH A DEVICE SERVER	29
FIGURE 18: INTEGRATION OF THE REAL-TIME ANIMATION AND STREAMING SYSTEM IN THE T-SYSTEMS TESTBED	33
FIGURE 19: APPLICATION PIPELINE OF THE HOLOGRAPHIC HUMAN-TO-HUMAN COMMUNICATION	35
FIGURE 20: NEURAL RE-RENDERING FOR REAL-TIME PERFORMANCE CAPTURE SYSTEMS [4]	38
FIGURE 21: IMPLEMENTED FRAMEWORK FOR LIVE TELEPORTATIO.....	40
FIGURE 22: COMPONENTS OF THE REAL-TIME ANIMATION AND STREAMING OF REALISTIC AVATARS APPLICATION.....	43
FIGURE 23: HUSKY ROBOT	46
FIGURE 24: SIMPLIFIED CONTROL CENTER	46
FIGURE 25: END-TO-END SECURITY	48
FIGURE 26: TRUST RELATIONSHIP IN CONFIDENTIAL COMPUTING	50
FIGURE 27: BASIC MEASURED START ON LINUX.....	51
FIGURE 28: MEASURED START WITH FULLY ENCRYPTED DISK IMAGES	52
FIGURE 29: APPLICATION EXAMPLE - SECURE CERTIFICATE PROVISIONING	54
FIGURE 30: SETUP OF THE GUEST VM	55
FIGURE 31: MANAGING THE GUEST VM	58

FIGURE 32: PLATFORM ARCHITECTURE FOR SUBJECTIVE QUALITY ASSESSMENT	61
FIGURE 33: CONFIGURATION CONTROL PANEL AND PC OBJECTS	62
FIGURE 34: SUBJECTIVE TEST CONFIGURATION UI	63
FIGURE 35: TEST OBJECTS IN 8I VOXELIZED FULL BODIES DATABASE	64
FIGURE 36: IMMERSIVE RATING SLIDER WITHIN THE USER INTERFACE OF THE HOLOLENS 2 AS USED DURING THE SUBJECTIVE TESTS	65
FIGURE 37: QUALITY RATINGS FOR DIFFERENT QUALITY LEVELS AND QUALITY SWITCHES	69
FIGURE 38: AVERAGE QUALITY RATINGS FOR DIFFERENT DISTANCES	69
FIGURE 39: AVERAGE QUALITY RATINGS OF PARTICIPANTS. IT SHOULD BE NOTED THAT THE SEQUENCE QII IN TASK 1 IS EQUIVALENT TO QI_D3 ($I \in \{1, 2, 3\}$) IN TASK 2, ENCODED AT QUALITY QI AND VIEWED AT 5 M.	70
FIGURE 40: CYBERSICKNESS LEVELS OF THE PARTICIPANTS	70
FIGURE 41: OBJECTS' IMMERSION LEVELS	71
FIGURE 42: PERCEIVED MOS (FROM OUR SUBJECTIVE TEST) VERSUS PREDICTED MOS USING THE GRADIENT BOOSTING REGRESSOR. THE RED LINE REPRESENTS THE Y = X LINE.	72
FIGURE 43: FEATURE IMPORTANCE SCORES OF INPUT FEATURES IN GRADIENT BOOSTING REGRESSOR.	73
FIGURE 44: CISCO ASA LOGIN SCREEN	81
FIGURE 45: CISCO SECURE CLIENT DOWNLOAD PAGE	82
FIGURE 46: VPN CONNECTION WITH CISCO SECURE CLIENT	83
FIGURE 47: NAS GUI WITH FILESTATION APPLICATION	85
FIGURE 48: IMPORT OF OVA FILE	86
FIGURE 49: RESULT OF OVA IMPORT	87
FIGURE 50: NAT NETWORK IN VIRTUALBOX	88
FIGURE 51: CHANGE VM OS AND VERSION	89
FIGURE 52: STORAGE TAB IN VM SETTINGS	90
FIGURE 53: HARD DISK SELECTION IN VIRTUALBOX	91
FIGURE 54: VIRTUAL DISK FILE TYPE DIALOG	92
FIGURE 55: VIRTUAL DISK ALLOCATION DIALOG	93
FIGURE 56: VIRTUAL DISK SIZE DIALOG	94
FIGURE 57: RESULT OF VIRTUAL DISK CREATION	95
FIGURE 58: VIRTUALBOX NETWORK CONFIGURATION	96
FIGURE 59: HELLO WORD APPLICATION IN THE TRUSTED GUEST VM	104



LIST OF TABLES

TABLE 1: USE CASE: HOLOGRAPHIC COMMUNICATION SPECIFICATIONS..... 31

**TABLE 2: USE CASE: REAL-TIME ANIMATION AND STREAMING OF REALISTIC AVATARS
34**

**TABLE 3: USE CASE HOLOGRAMS: DEVELOPMENT/INTEGRATION AND PERFORMANCE
39**

**TABLE 4: USE CASE MULTI-SOURCE: DEVELOPMENT/INTEGRATION AND
PERFORMANCE..... 41**

TABLE 5: USE CASE AVATAR: DEVELOPMENT/INTEGRATION AND PERFORMANCE 45

TABLE 6: USE CASE ROBOT: DEVELOPMENT/INTEGRATION AND PERFORMANCE 47

TABLE 7: BITRATES IN MBIT/S OF DIFFERENT QUALITY LEVELS OF THE PC OBJECTS 64

TABLE 8: NOTATION AND DESCRIPTION OF THE TEST SEQUENCES 65

TABLE 9: STRUCTURE OF RATING SCORES DATABASE 66

TABLE 10: STRUCTURE OF QUESTIONNAIRE ANSWERS..... 67

**TABLE 11: PERFORMANCE OF MACHINE LEARNING MODELS IN PREDICTING THE QOE
OF POINT CLOUDS IN AR ENVIRONMENTS. THE BOLD ENTRY SIGNIFIES THE BEST
PERFORMANCE..... 72**

**TABLE 12: PERFORMANCE OF THE ORIGINAL P.1203 MODE 0 AND OUR FINE-TUNED
P.1203 WITH TRAINING AND VALIDATION DATASET. 73**

ABBREVIATIONS

5G CPE	5G Customer Premises Equipment
5G NR	5G New Radio
5G SA	5G Stand Alone
5G/6GIC	5G/6G Innovation Centre
ACME	Automatic Certificate Management Environment
ANOVA	Analysis of Variance
API	Application programming interface
ASA	Adaptive Security Appliance
AR	Augmented Reality
BBU	Broadband Unit
CO	Confidential
CTC	Common Test Conditions
CSV	Comma-Separated Values
C-V2X	Cellular Vehicle-to-Everything
DCMS	Department for Digital, Culture, Media, and Sport
DT	Deutsche Telekom
EFI	Extensible Firmware Interface
ESA	European Space Agency
FoV	Field of View
G-QP	Geometry Quantization Parameter
GUI	Graphical User Interface
HAS	HTTP Adaptive Streaming
HEVC	High Efficiency Video Coding
HMD	Head-Mounted Display
HSD	Honestly Significant Difference
HW	Hardware
ID	Identifier



IP	Internet Protocol
ITU	International Telecommunication Union
JANET	Joint Academic Network
JISC	Joint Information Systems Committee (UK)
KVM	Kernel-based Virtual Machine
LTE	Long-Term Evolution
MEC	Multi-access Edge Computing
MOS	Mean Opinion Score
MPEG	Moving Picture Experts Group
MR	Mixed Reality
MRTK	Mixed Reality Toolkit
MSE	Mean Squared Error
NAS	Network Attached Storage
NTN	Non-Terrestrial Network
OS	Operating System
OVA	Open Virtualization Appliance
OVF	Open Virtualization Format
OVMF	Open Virtual Machine Firmware
PBKDF2	Password-Based Key Derivation Function 2
P2P	Peer-To-Peer
PC	Point cloud
PCC	Point cloud compression
PLCC	Pearson Linear Correlation Coefficient
PLMN	Public Land Mobile Network
PSP	Platform Secure Processor
PU	Public
QCOW2	QEMU-Format Copy On Write 2
QMP	Qemu Management Protocol

QoE	Quality of Experience
QP	Quantization Parameter
RAM	Random Access Memory
RAN	Radio Access Network
RAT	Radio Access Technology
RGB	Red Green Blue
RE	Restricted
REST	Representational State Transfer
RMSE	Root Mean Squared Error
SA	Standalone
SEV	Secure Encrypted Virtualization
SDK	Software Development Kit
SDN	Software-Defined Networking
STUN	Session Traversal Utilities for NAT
SME	Small and Medium Enterprises
SNMP	Simple Network Management Protocol
SRCC	Spearman's Rank Correlation Coefficient
SSH	Secure Shell
T-QP	Texture Quantization Parameter
TAR	Tape Archive
TCB	Trusted Computing Base
TCP	Transmission Control Protocol
TMC2	Test Model Category 2
3D	Three-dimensional
TLS	Transport Layer Security
2D	Two-dimensional
UAV	Unmanned Aerial Vehicle
UEFI	Unified Extensible Firmware Interface

UI	User Interface
UKRI	UK Research and Innovation
URL	Unified Resource Location
URLLC	Ultra-Reliable Low-Latency Communications
UNI-KLU	University of Klagenfurt
VM	Virtual Machine
VMM	Virtual Machine Manager
VPN	Virtual Private Network
VR	Virtual Reality
WebRTC	Web Real-Time Communication
YAML	YAML Ain't Markup Language

1 INTRODUCTION

Telepresence can be seen as the next generation of communication applications that will significantly enrich the human-to-human and human-to-machine experience, blurring the boundaries between the physical and virtual worlds. Such systems are expected to fundamentally change the way people communicate and collaborate with each other in various sectors such as education, training, entertainment, retail, healthcare, manufacturing and many others. The further development of telepresence services will contribute significantly to increasing society's resilience to environmental disasters, boosting industrial productivity and improving energy efficiency, thanks to changes in people's lifestyles and work habits.

Until now, telepresence solutions have been high-end, expensive, and quite conventional audio and video conferencing systems. However, the last decade has seen intensive research and development on VR/AR/XR technologies and applications that have significantly improved the state of the art. This has led to interesting immersive telepresence and/or collaboration systems, some of which are still in the research stage. Due to their complexity, cost, data compression, and bandwidth requirements, these solutions have not scaled yet.

The mission of the SPIRIT project is to put real-time immersive telepresence into practise by researching, integrating, and further developing state-of-the-art immersive telepresence technologies, components and platforms to create Europe's first multi-site and interconnected framework able to support the operation of heterogeneous collaborative telepresence applications at large scale.

1.1 PURPOSE OF THE DOCUMENT

This report is the first version of the tested and validated SPIRIT platform with all currently available components and interfaces. It is one part of a bundle of deliverables providing insight in the project's use cases, requirements, architecture, available components, upcoming enablers, and the integrated SPIRIT platform:

- D2.1 "Use Case Requirements, System Architecture and Interface Definition (First Version)" [1]
- D3.1 "Innovation Platform Enablers (First Version)" [2]
- **D4.1 "SPIRIT Platform (First Version)" [3]**

On the one hand it reflects the ongoing work within the project especially the tasks:

- **Platform development** - implement and/or enhance all platform components and interfaces.
- **Technical integration and validation** – integrate the components of the different project partners to form an overall platform demonstrator.
- **Use case development and integration** - ensures the development of the project-designated use cases and their integration with the underlying platform available.
- **User experience evaluation and usability validation** - provide and test quality of experience (QoE) metrics and procedures to assess immersive telepresence solutions and support of Open Call experiments.

On the other hand, it provides the applicants of the first Open Call with insights regarding the components available to enhance their use cases, and identify complementary components to the SPIRIT platform, which they may want to provide to the overall SPIRIT goal.

There will be two more iterations of this document. In the next iteration, further enhancements and improvements added to the first version of the SPIRIT platform will be documented in a second version and at the end of the project a report about the final version of the SPIRIT platform is planned.

1.2 STRUCTURE OF THE DOCUMENT

The sections of the report at hand are organised in the following manner:

The chapter 2 “Testbeds”, provides an overview of the testbeds in Surrey and Berlin, e.g. the available 5G network infrastructure and the computing power of the EdgeCloud located there.

The chapter 3 “Platform components” lists partner components that were integrated into the testbeds during the project and documents the findings of this integration process.

The chapter 4 “Implemented Use Cases” covers in detail the implementation of use cases, which are described in detail in [1]. On the one hand, the integration of the platform components into the use cases and the description of the necessary adjustments for a smooth integration into the testbeds are provided and, on the other hand, a look ahead to how these components might find use in various scenarios, such as third-party applications, is presented.

The chapter 5 “Security Development and Technical Integration” specifies an approach for the deployment of Confidential Computing-protected virtual machines (VM) on bare metal servers (either on-premise or in the cloud) where private keys for managing VMs never leave the data owner’s control and specifically do not need to be uploaded to cloud provider systems. Appendix A is a manual to setup and make use of the Confidential Computing.

The chapter 6 “Quality of Experience Evaluation” presents the subjective tests conducted by partners to assess the impact of quality, quality switching, viewing distance, and content characteristics on the perception of point clouds in AR environments. The output of this work includes (i) a platform for subjective quality assessment in AR environments, (ii) a dataset of rating scores that can be used for training and validating future QoE models as well as the results (findings) of the subjective tests that produced these rating scores, and (iii) a machine learning based QoE model. In appendix B is the Subjective Test platform described.

The chapter 7 “Conclusions” concludes the document.

2 TESTBEDS

This section provides an overview of the testbeds in Surrey and Berlin, e.g. the available 5G network infrastructure and the computing power of the EdgeCloud located there.

2.1 SURREY

2.1.1 General Information on University of Surrey Testbed

The UK site facility is managed by the 5G/6G Innovation Centre at the University of Surrey, in Guildford (Surrey). The testbed offers 5G infrastructure including campus-wide 4G/ Long-Term Evolution (LTE) and 5G/New Radio (NR) based radio access network, virtualised 4G and 5G core network, managed Software-Defined networking (SDN) and fibre external connectivity, as well as Unmanned Aerial Vehicle (UAV) and satellite systems.

The testbed supports domestic (UK) projects as well as European and International projects. Recent examples thereof are European Space Agency (ESA) 5G-TINA and SUNRISE 5G Pilot, Department for Digital, Culture, Media, and Sport (DCMS) Flex5G and FONRC TUDOR and UK Research and Innovation (UKRI) UK India (UKI-FNI). Small and medium enterprises (SMEs) are also in a partnership with the centre, in order to use the testbed to test different aspects of 5G-and-beyond technologies, such as Non-Terrestrial Network (NTN) communications, high accuracy time synchronization, network slicing, management and orchestration, flexible network functions disaggregation and energy efficiency.

The 5G/6GIC site offers a multi-Radio Access Technology (RAT) radio access network covering the University Surrey campus over four square kilometres and several off-campus locations where coverage is provided. It features both outdoor and indoor base radio units, supporting a mix of 4G/LTE and 5G/NR technologies, as shown in Figure 1. These base stations are used to showcase both 5G standalone (SA) and non-standalone (NSA) mobile network deployment scenarios and are currently used to support projects and research activities requiring traditional Radio Access Network (RAN) deployments.

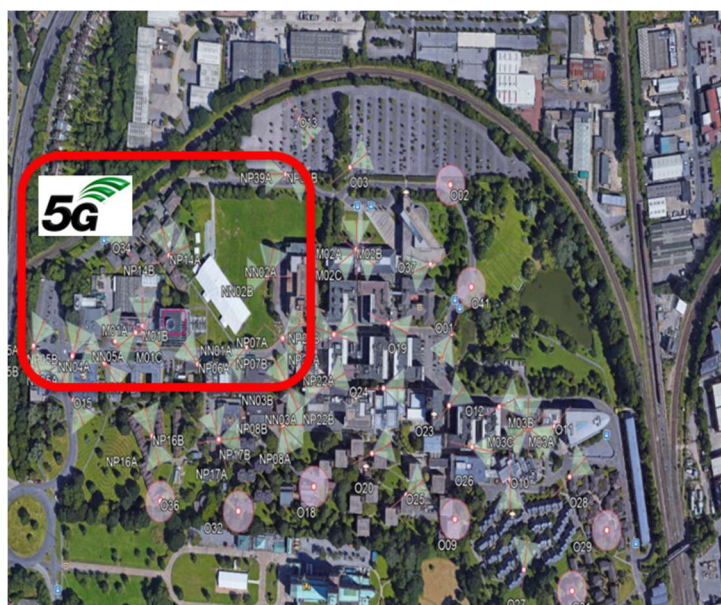


FIGURE 1: 5G TESTBED IN UNIVERSITY OF SURREY

2.1.2 Testbed Infrastructure

In the 5G testbed, the outdoor coverage consists of 40 LTE sites with 23 centralised BBUs hosted in 5G/6GIC, supporting 2.6GHz band 28 and 38 for a total of 120 cells. Furthermore, there are 3 outdoor NR sites in the 3.5GHz band n78 for a total of 9 cells and another 3 sites in the 3.7GHz band n77 consisting of 3 cells, with respectively, 3 and 2 centralised BBUs hosted in machine rooms in 5G/6GIC. Concerning infrastructure, as shown in Figure 2, the UK trial network hosts two machine rooms hosting physical computing infrastructure and broadband units (BBUs) for both 4G and 5G, as well as both hardware and software-defined networking equipment. The UK trial network is interconnected with several UK and international sites via JISC's JANET network, as well as via a satellite link. Finally, the UK trial network features additional equipment such as connected autonomous vehicles for C-V2X scenarios, UAV systems for scenarios of emergency pop-up networking, a prototype URLLC integrated setup used to demonstrate low latency use cases of up to 6K users.

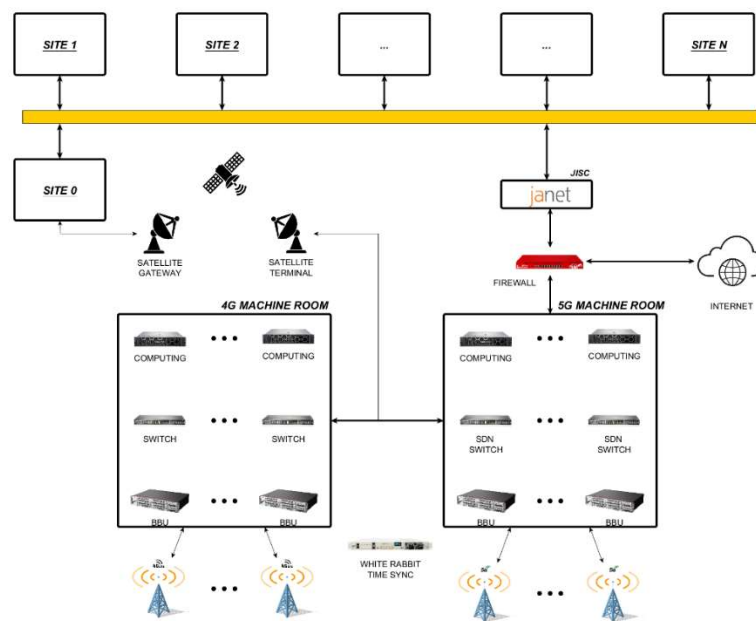


FIGURE 2: INFRASTRUCTURE AND CONNECTIVITY

Regarding the Holographic server and client connecting via 5g network, Figure 3: Holographic server and client connected via 5G network displays their hardware and software components. At left side, the holographic server application is deployed on 5G MEC server which is a Dell Precision 7960 XCTO base machine. In this machine, it has two Nvidia GeForce RTX A6000 video card with 48 GB memory, a system memory at 256GB and three hard drive disks. On this machine, Windows 11 Pro for workstations is installed with the CUDA library to provide hardware acceleration to the application server software. On system containers like Ubuntu can be enabled in the embedded containers, with same ability to access video card resources. This machine is directly connected to 5G network, and the 10G network interface can effectively receive raw hologram frame from clients. At the client side, two Dell Alienware R15 PCs are deployed with livescan3d client application to fetch raw hologram frame from camera. The CPU is Intel Core i9 13900KF 24 cores. The video card is NVIDIA GeForce RTX 4090 with 24GB memory. The system memory is 32 GB and hard disk spaces is 1TB + 1TB. With the local connectivity to 5G Customer Premises Equipment (5G CPE), the raw data can be streamed to 5G MEC server.

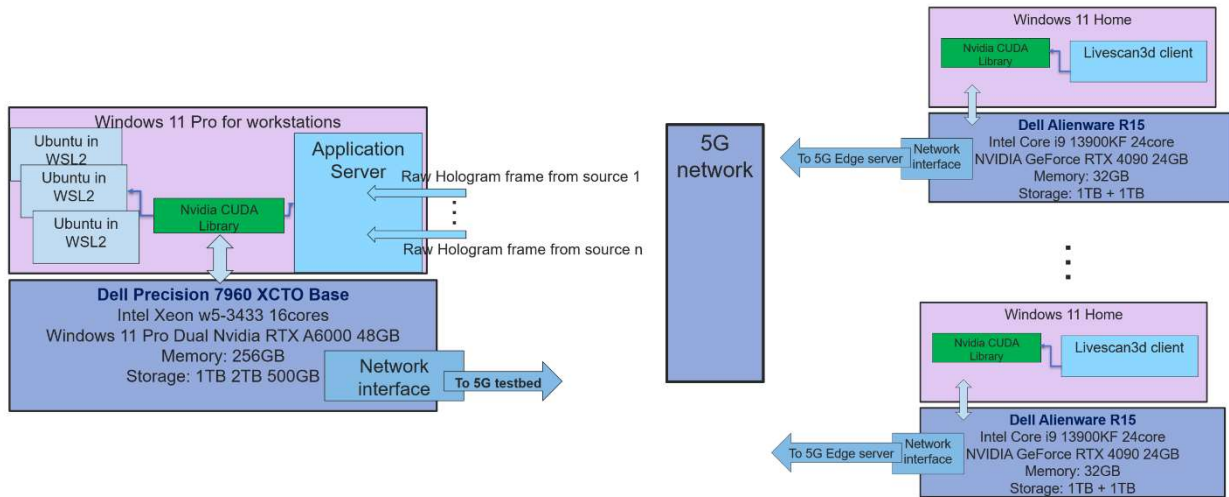


FIGURE 3: HOLOGRAPHIC SERVER AND CLIENT CONNECTED VIA 5G NETWORK

2.1.3 Onsite Use Cases

During the SPIRIT Project the University of Surrey has worked together with the other SPIRIT partners to bring the use case “multi-source live teleportation with MEC support” use cases to the testbed and make them available for open call participants. Detailed information can be found in the use case descriptions of D2.1 “Use Case Requirements, System Architecture and Interface Definition (First Version)” [1].

2.1.4 Onsite Collaborative Opportunities

There are two ways to collaborate with Surrey’s 5G testbed. 1) Deploy a container-based application at a specific 5G core slice with a MEC server to utilize the GPU resources, and 5G access and a public internet connection. 2) Deploy a standalone application and just use the 5G connectivity for the user to access the public internet. For the first case, an executable application server program can be deployed in one of the containers. The user device (e.g., 5G phone or HoloLens with 5G phone tethering) of the application which is the content receiver then can connect to 5G radio or another public Internet address to exchange application messages and receive application data from the container server. In the second case, one or multiple devices can access the public Internet through the 5G radio network and can therefore connect to any remote application server deployed at the public internet.

2.2 BERLIN



FIGURE 4: LOCATION OF DEUTSCHE TELEKOM TESTBED

2.2.1 General Information Deutsche Telekom Testbed

The Deutsche Telekom 5G Testbed is located in Berlin, Germany at the “Siemensstadt Square” district where Deutsche Telekom collaborates with the Werner von Siemens Centre and various other partners coming from industrial, public and educational sectors. The research factory is situated in the same old building as the Siemens dynamo factory, which started its production in the early 1900s and is still active up to today. Many different partners experiment and cooperate on innovative topics right next to the actual factory. We are looking into a wide range of topics, including new and innovative manufacturing techniques, advanced transportation and mobility solutions, the shift towards more sustainable energy sources and many other interesting areas.



FIGURE 5: LOCATION OF TESTBED IN BERLIN

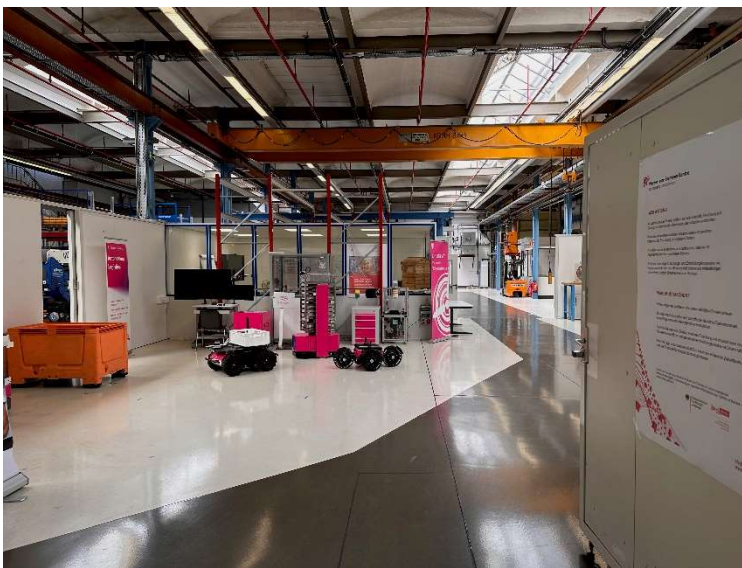


FIGURE 6: LAB FLOOR WERNER VON SIEMENS CENTER

2.2.2 Testbed Infrastructure

Together with the SPIRIT Project, Deutsche Telekom is opening its Future Factory for partners. The site offers an area of around 500 m² outdoor space and 1000 m² indoor space. The indoor area as well as the outdoor area is covered by a private 5G Standalone Network. The 5G network is configured for the 3.7- 3.8 GHz industrial spectrum and has been approved by the German Federal Network Agency. The virtualized 5G core supports the 3GPP standard up to Release 16. The indoor space is also partly covered with WIFI5 and WIFI6 and can be used to connect devices that aren't capable of 5G. Computing power is provided by an edge server that is located on premise and is connected via fibre to the 5G Network. The server runs containerized applications in a Kubernetes cluster. There are two Nvidia T4 16GB Graphic Cards, 96 GB of RAM and 24 Cores (3 x Intel(R) Xeon(R) Gold 6226R CPU @ 2.90GHz) available in the cluster that are being shared between all tenants.

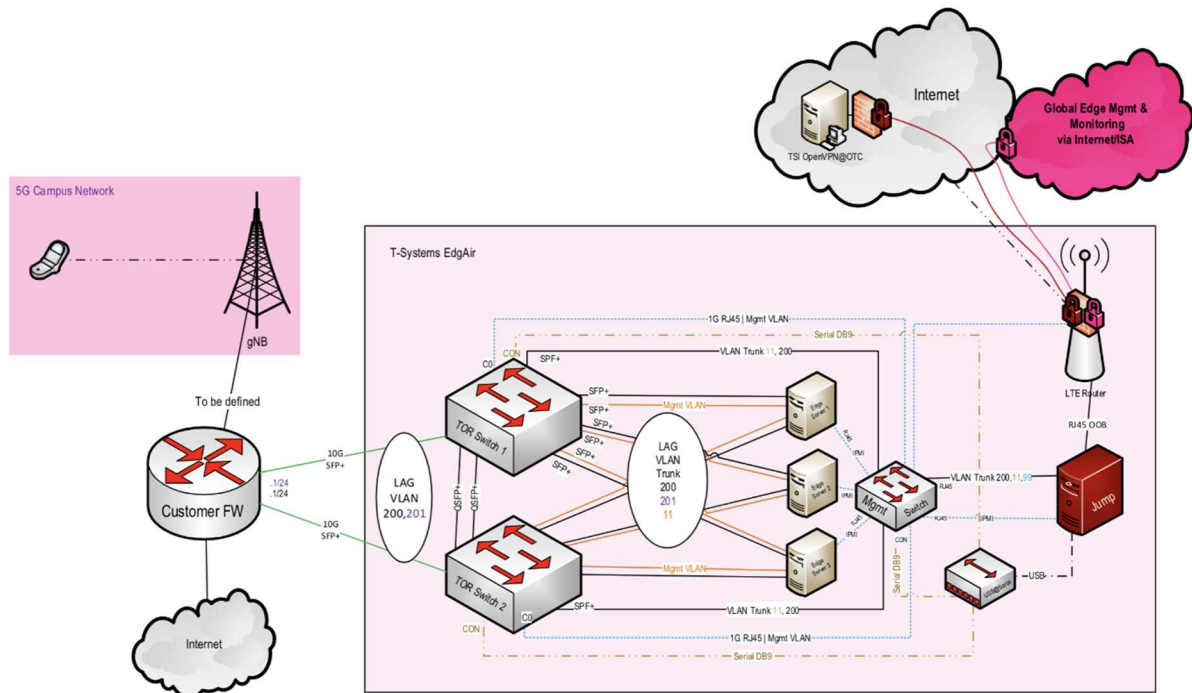


FIGURE 7: NETWORK INFRASTRUCTURE

2.2.3 Onsite Use Cases

During the SPIRIT Project the Deutsche Telekom has worked together with the other SPIRIT partners to bring three contributions and use cases to the testbed and make them available for open call participants:

- Real-Time Animation and Streaming of Realistic Avatars
- Holographic Human-to-Human **Communications**
- Network aware Kubernetes Scheduler

Furthermore the “Distributed Steering of Autonomous Mobile Robots (AMRs)” use-case is exclusively available in the Deutsche Telekom Testbed. Detailed descriptions of these use cases can be found in the use case descriptions of D2.1 “Use Case Requirements, System Architecture and Interface Definition (First Version)” [1].

2.2.4 Onsite Collaborative Opportunities

Participants can deploy their software on our local edge cloud. To do so they get a dedicated access to the Rancher UI, our Kubernetes management platform, to remotely deploy and manage their applications in the cluster. The Platform is accessible from the internet and gives partners the opportunity to deploy their software without the need of being onsite. The Platform uses multitenancy for ensuring both privacy and security. For this we use namespaces, which are divided into Rancher projects. Each project contains users divided by access levels within the project (we can configure an individual set of permissions or choose from predefined: Owner, Member, Readonly). Access to the storage is also possible only within the namespace\project.

Also for each project we can limit the allocation of resources (CPU, RAM, etc.).

For testing and collaboration, participants of the SPiRiT Project will have the possibility to make appointments for onsite visits. In general, except when otherwise stipulated, Participants are supposed to bring their own devices to the testbed to test their use case. We offer three ways to connect these devices to our network:

- ➡ 5G: We will provide a SIM Card that is configured for the local 5G network
- ➡ WiFi: We help connecting to an access-controlled WiFi
- ➡ LAN: We will provide an ethernet port to connect to the cluster

As not all 5G devices are able to connect to our private 5G network, we can provide a list of devices that are capable of doing so.

There is a managed firewall active inside the cluster, which is why network traffic needs to be approved and configured.

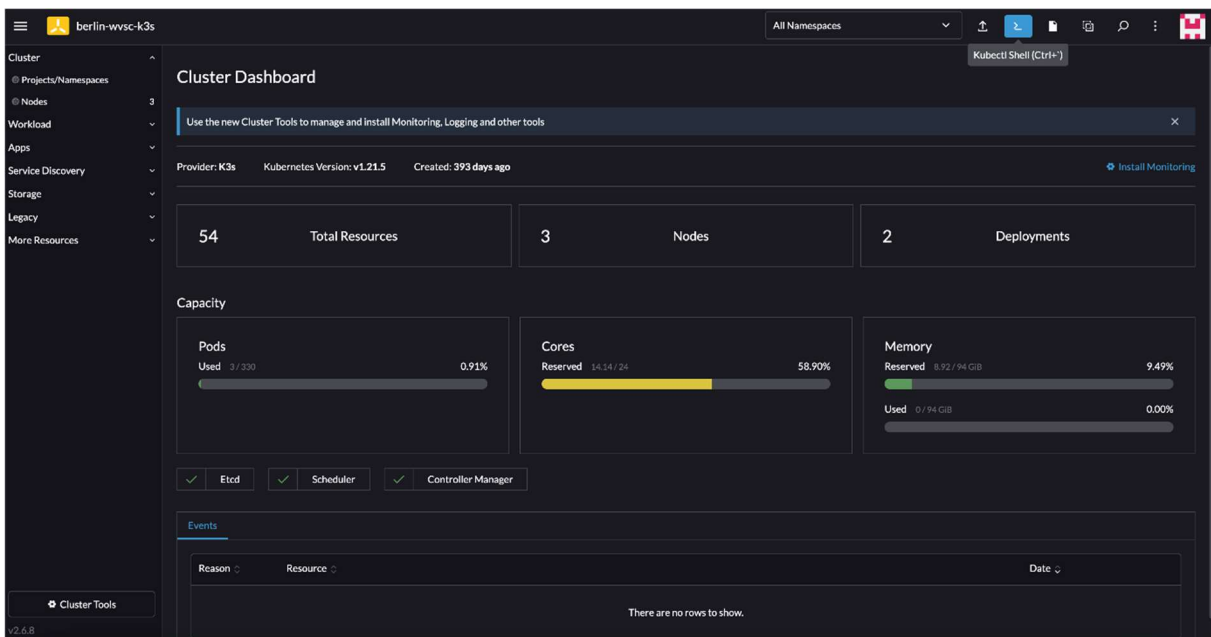


FIGURE 8: KUBERNETES MANAGEMENT PLATFORM

3 PLATFORM COMPONENTS

The section lists partner components that were integrated into the testbeds during the project and documents the findings of this integration process.

3.1 LIVE MULTI-SOURCE HOLOGRAPHIC STREAMING

This section is about the components supporting applications using live teleporting people from remote internet locations to a common virtual space of the audience such that the audience can have the immersive and multisensory perception that everyone is located in the common physical scene.

3.1.1 Deployment of the Frame Render and Synchronization Function

There are two key components for enabling a multi-source live volumetric streaming application: a frame synchronization function and a RESTful interface to accept runtime parameter settings and queries (as shown in the orange block in Figure 9). The frame synchronization function is responsible for rendering holographic frames from multiple independent sources. For frames from different sources, the MEC server can distinguish them based on their source IP addresses and record their timestamps separately. Based on a predefined synchronization threshold (e.g., 30ms) and the real-time timing of each source, the MEC server can perform operations such as pairing, buffering, and discarding of multi-source frames to ensure that the synthesized frame exhibits optimal performance. Meanwhile, this frame synchronization function can also be configured through RESTful interfaces. The frame synchronization threshold and required key performance metrics (e.g., throughput, playback latency) can be set using the POST method and queried using the GET method through an authorized network management interface. Figure 9 displays the application of the frame synchronization function on Surrey's testbed. The rendering function can directly call the NVIDIA Compute Unified Device Architecture (CUDA) library to utilize the graphic card resources of the platform, enabling efficient computation and display of multi-source frames on this platform.

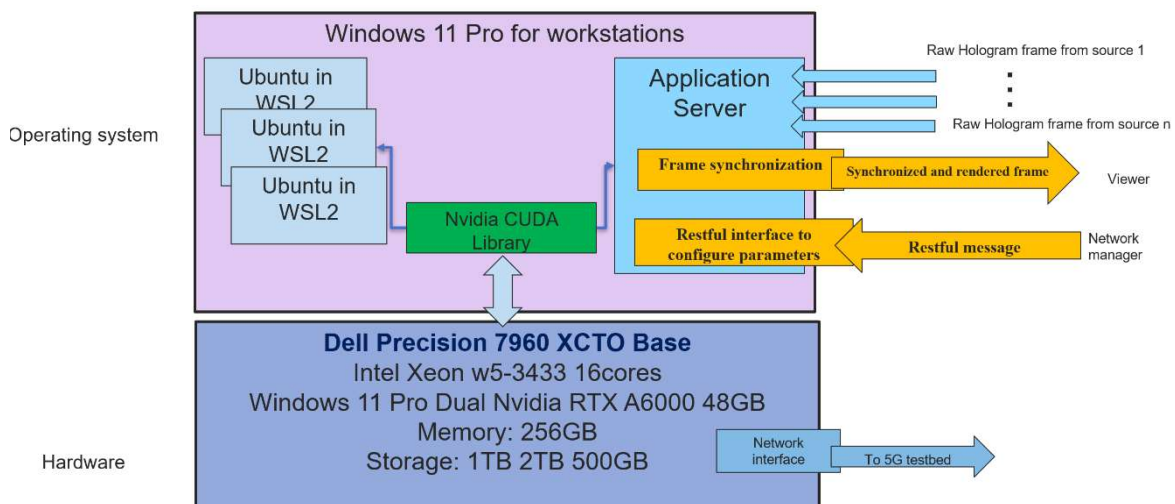


FIGURE 9: DEPLOYMENT OF FRAME PRODUCTION AND SYNCHRONIZATION FUNCTION ON SURREY'S PLATFORM

3.1.2 Validation

Figure 10 shows a captured frame from a demonstration of Surrey live multi-source holographic platform with the frame render and synchronization function. There are two sources

located at location A and B, with dedicated cameras to capture different people in a live streaming manner. The virtual space in the middle is the real-time screen projected from the edge server. It shows the merged and synchronized frame, which can validate the efficacy of the remote production and frame synchronization function.



FIGURE 10: VALIDATION OF THE FRAME PRODUCTION AND SYNCHRONIZATION FUNCTION ON SURREY'S PLATFORM

3.2 NETWORK-AWARE RESOURCE SCHEDULER (DIKTYO)

To avoid monolithic architectures, telepresence solutions typically consist of multiple individual microservices or components, in the form of containers, that together represent a service function chain. In order to ensure that the deployments of such chains still meet the latency and throughput requirements of the application, resource management and orchestration platforms should be made aware of the network characteristics as well as of computational resource usage. Vanilla Kubernetes (K8s) deployments do not take networking characteristics into account and primarily focus on CPU and RAM optimization.

The Diktyo network-aware scheduler has been developed to alleviate this problem, as described in more detail in deliverable D3.1. It has been proven to improve network throughput considerably for several benchmarking applications.

Diktyo supports easy deployment in a Kubernetes (K8s) cluster as an additional scheduler in the system without needing to remove the default scheduler. A helm-chart has been developed to automatically deploy all Diktyo components, available here:

➔ <https://github.com/diktyo-io/helm-chart/tree/main>.

Documentation with required steps and verification that all pods are running properly as seen in Figure 11 is provided. Different versions of Diktyo exist since K8s v1.24 in the K8s scheduling community available as different releases:

➔ <https://github.com/kubernetes-sigs/scheduler-plugins/releases>.

The Diktyo scheduler can be configured with different parameters as shown in Figure 12. Additional information on how to deploy additional schedulers in K8s clusters can be found in the K8s scheduling community:

➔ <https://github.com/kubernetes-sigs/scheduler-plugins/blob/master/doc/install.md>

```
$ kubectl get deploy -n diktyo
NAME                    READY   UP-TO-DATE   AVAILABLE   AGE
appgroup-controller    1/1     1             1           22s
diktyo-scheduler       1/1     1             1           22s
networktopology-controller 1/1     1             1           22s
scheduler-plugins-controller 1/1     1             1           22s
```

FIGURE 11: DIKTYO SCHEDULER RUNNING PROPERLY

Parameter	Description	Default
<code>scheduler.name</code>	Scheduler name	<code>diktyo-scheduler</code>
<code>scheduler.image</code>	Scheduler image	<code>registry.k8s.io/scheduler-plugins/kube-scheduler:v0.26.7</code>
<code>scheduler.leaderElect</code>	Scheduler leaderElection	<code>false</code>
<code>scheduler.weight</code>	Scheduler Weight (Score plugin)	<code>5</code>
<code>scheduler.replicaCount</code>	Scheduler replicaCount	<code>1</code>
<code>controller.name</code>	Controller name	<code>scheduler-plugins-controller</code>
<code>controller.image</code>	Controller image	<code>registry.k8s.io/scheduler-plugins/controller:v0.26.7</code>
<code>controller.replicaCount</code>	Controller replicaCount	<code>1</code>
<code>appGroupController.name</code>	appGroupController name	<code>appgroup-controller</code>
<code>appGroupController.image</code>	appGroupController image	<code>jpedro1992/appgroup-controller:v1.0.3-alpha</code>
<code>appGroupController.replicaCount</code>	appGroupController replicaCount	<code>1</code>
<code>networkTopologyController.name</code>	networkTopologyController name	<code>networktopology-controller</code>
<code>networkTopologyController.image</code>	networkTopologyController image	<code>jpedro1992/networktopology-controller:v1.0.3-alpha</code>
<code>networkTopologyController.replicaCount</code>	networkTopologyController replicaCount	<code>1</code>
<code>plugins.namespaces</code>	Plugins namespaces by default	<code>["default"]</code>
<code>plugins.weightsName</code>	Plugins weightsName by default	<code>"NetperfCosts"</code>
<code>plugins.networkTopologyName</code>	Plugins networkTopologyName by default	<code>"nt-cluster"</code>

FIGURE 12: DIKTYO SCHEDULER CONFIGURATION PARAMETERS

3.2.1 Integration with CloudNativeLab

During the SPIRIT project, Diktyo has been successfully integrated into the CloudNativeLab at IMEC-IDLab:

➔ <https://practicum.cloudnativelab.ilabt.imec.be/>.

CloudNativeLab is a testbed for experimenting with K8s and cloud native technologies. With only a few clicks, experimenters can spin up their own K8s cluster in the IMEC-IDLab data center (Figure 13 and Figure 14).

Diktyo has been integrated into K8s clusters created with CloudNativeLab and can be used to deploy applications in the system by changing the `schedulerName` field in deployment files (Figure 15) as shown in the K8s documentation:

➔ <https://kubernetes.io/docs/tasks/extend-kubernetes/configure-multiple-schedulers/>.

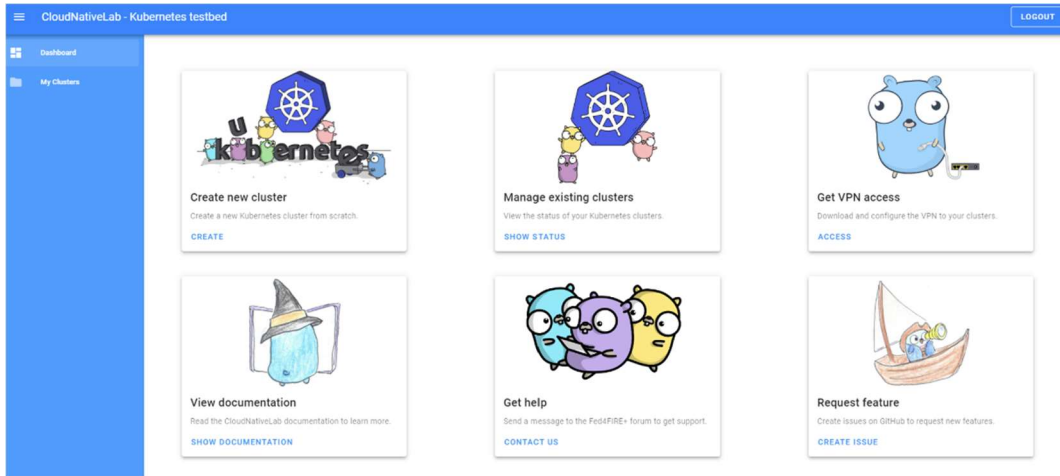


FIGURE 13: CLOUDNATIVELAB MAIN DASHBOARD

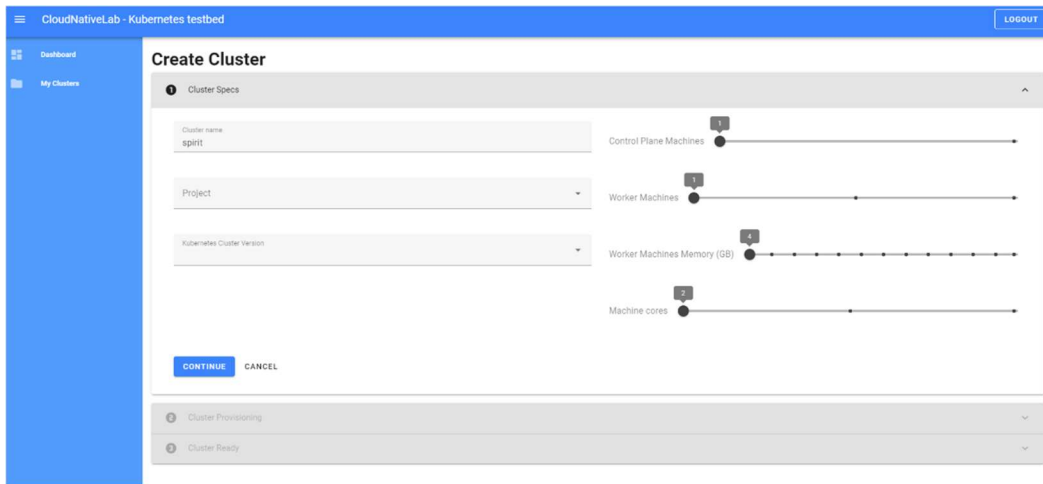


FIGURE 14: CLOUDNATIVELAB K8S CLUSTER CREATION PAGE

```

admin/sched/pod2.yaml
apiVersion: v1
kind: Pod
metadata:
  name: annotation-default-scheduler
  labels:
    name: multischeduler-example
spec:
  schedulerName: default-scheduler
  containers:
  - name: pod-with-default-annotation-container
    image: registry.k8s.io/pause:2.0
    
```

FIGURE 15: DEPLOYMENT EXAMPLE THAT SPECIFIES SCHEDULER FOR POD DEPLOYMENT

3.2.2 Integration with Berlin Testbed

During the SPIRIT project, integration efforts have taken place to test and validate Diktyo in the T-Systems testbed. The integration has been challenging since the T-Systems testbed currently supports K8s v1.21 and the most recent version of Diktyo is available for K8s v1.26. The mismatch of K8s API versions has hindered the complete integration of the most recent version of Diktyo in the T-Systems testbed.

Nonetheless, an older version of Diktyo (v1.22) has been currently deployed and successfully integrated into the T-Systems testbed, being able to deploy microservices in the system and providing the required functionality to support experimenters with network-awareness when scheduling application containers.

3.3 HOLOGRAPHIC HUMAN-TO-HUMAN COMMUNICATION

The first version of the SPIRIT platform aims to support human-to-human communication use cases for immersive telepresence applications, as described in D2.1 [1]. In addition, within the SPIRIT project the goal is to incorporate innovative platform enablers presented in D3.1 [2] to enhance the immersive telepresence experience.

Prior to exploring technical considerations of the holographic human-to-human communication use case, more specifically holographic communication, it is crucial to emphasize the profound significance of capturing authentic experiences.

For example, in professional settings, holographic communication transforms collaborative endeavors by offering lifelike depictions of remote participants. The immersive nature of 3D representation fosters a collaborative environment where individuals can share ideas, engage in discussions, and collaborate on projects as if they were physically co-located. The ability to visualize the spatial arrangement of team members enhances the collaborative process, possibly contributing to more innovative and human-centric outcomes.

Figure 16 illustrates the deployed platform for holographic human-to-human communications:

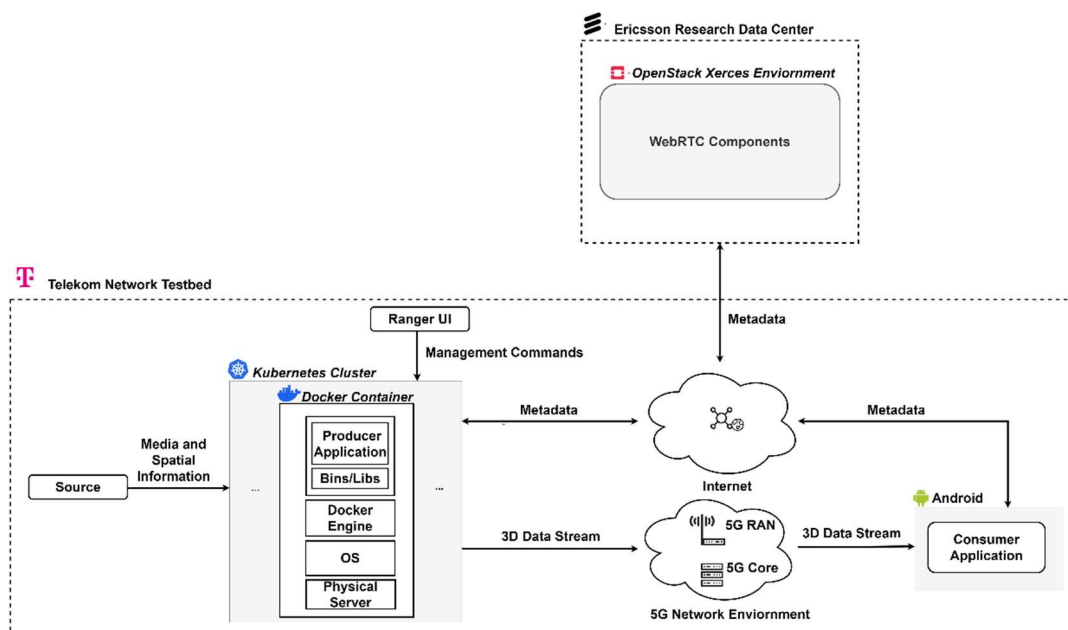


FIGURE 16: DEPLOYED PLATFORM FOR HOLOGRAPHIC HUMAN-TO-HUMAN COMMUNICATIONS

Within the SPIRIT project, the components of the application platform supporting holographic human-to-human **communications** by Ericsson is implemented into the network testbed provided by Deutsche Telekom. The procedure mainly involves the integration of the components primarily responsible for data processing into the network testbed.

This process entails importing the producer-side application into an Edge Cloud system detailed in Section 2.2.2, while maintaining the consumer application on a mobile end device. Leveraging the computational capabilities of the Edge Cloud system facilitates expedited data processing by the producer application software, thereby possibly reducing end-to-end latency. This approach is particularly significant for digital human-to-human **communication** use cases with real-time requirements.

The crucial hardware components of the application platform will be provided to third parties, this includes:

- Depth camera
- Edge Cloud (Kubernetes cluster & Docker container)
- WebRTC Component(s)
- 5G Android mobile phone
- AR glasses

Other components that may be required to extend the application platform to support additional use cases must be provided by third parties.

The following sections describe the platform components provided by Ericsson and highlights some aspects of the integration procedure that are needed to implement the holographic human-to-human **communications** use case for the first version of the SPIRIT platform.

3.3.1 Common Prerequisites for Integration

Initially, it is essential to establish a common set of requirements for both the application platform and the network testbed for the integration procedure. The network testbed is designed to facilitate the deployment and execution of applications within a container-based environment. This environment enables seamless connectivity between diverse containers and provides access to tangible resources like GPU cards and computational power.

The producer application is containerised using Docker to integrate the application platform into the network testbed. This choice is motivated by Docker containers ensuring consistent, portable, and efficient software deployment. They encapsulate applications and dependencies, facilitating easy scaling, versioning, and streamlined management across diverse environments. The orchestration of these containers is managed by a Kubernetes Cluster, utilizing the Rancher management user interface for control.

A 5G network environment is in place to establish connections between SPIRIT platform devices, such as mobile phones, within a controlled network scenario.

3.3.2 Compatibility Test of Client Devices

To integrate the consumer application, the application is downloaded and installed on mobile end device equipped with a SIM card. The SIM card is provided by Deutsche Telekom and allows to connect the mobile end device to a controlled 5G test network within the Deutsche

Telekom testbed. It is noteworthy that the mobile end device must meet certain compatibility criteria. This requires knowledge about the public land mobile network (PLMN) identification number of the testbed and compatibility with augmented reality (AR) glasses, more specifically the XReal Light glasses used to develop and test the application. The latter criterion necessarily requires an Android operating system.

A range of Android test devices were available for us to assess connectivity with the 5G environment within the testbed. These included the Samsung Galaxy S20, Samsung Galaxy S8, Oppo Find X2 Pro, Oppo Find X3 and Motorola Edge +. Here, the Oppo Find X2 Pro and Oppo Find X3 phones were proven to be supported by the consumer applications of Ericsson. Tests on the consumer application implemented on the network testbed for other mobile phones are investigated.

In our compatibility test of suitable client devices, the Oppo Find X2 Pro mobile phone has been identified as consumer application device that satisfies both described compatibility criteria.

It is important to note, that the consideration of other phones is possible. However, mobile phones provided by third parties need to meet the described criteria. Therefore, we suggest selecting mobile end devices in consultation with Ericsson and Deutsche Telekom to provide a seamless integration of mobile end devices that have not been tested with the platform.

3.3.3 Integration Efforts

The holographic human-to-human **communications** use case demands a depth camera representing as source to capture media and spatial information of an object of interest. In this case the object of interest is represented by a human face or torso. The data obtained from the camera is used by the producer application to compute 3D representations in the form of holograms, that are streamed to the consumer application once a WebRTC connection has been set up exchanging meta data in signalling, as described in D3.1 [2].

It is crucial to highlight that the depth camera does only provide a USB 3.0 port and does not offer a network interface. Hence, a solution that bridges the connection between the depth camera and the Edge Cloud system is needed. Two solutions are considered:

1. Solution using a dedicated device server, as seen in Figure 17:

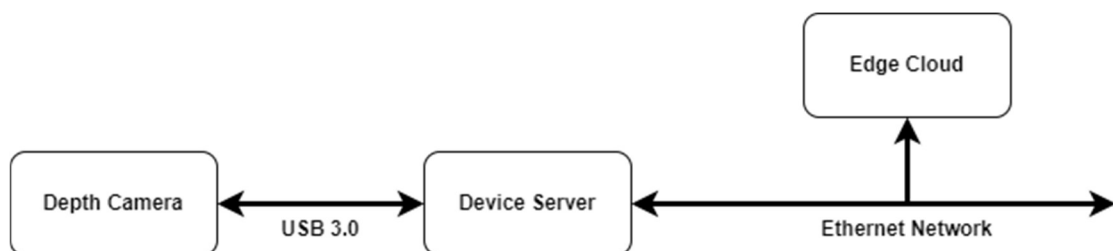


FIGURE 17: CONNECTIVITY BETWEEN DEPTH CAMERA AND EDGE CLOUD WITH A DEVICE SERVER ¹

The device server provides two USB 3.0 interfaces that are compatible with our depth camera hardware. In addition, the device server offers a network interface based on the IEEE 802.3 (1000BaseT, 100BaseTX, and 10 BaseT) standard, which can be used to connect the Edge cloud to the device server. The hardware and the pre-installed

¹ [INU-100](#)

software of the device server enable a data connection between depth camera and Edge Cloud system without additional implementation efforts on application or testbed side.

In addition, the device server comes with additional features such as freely selectable power supply between 12 V and 48 V, integrated change-over relay with automatic or event-controlled switching in the event of defined status changes (e.g. power or network failure) or manual switching via web browser, and numerous notification options in the event of faults (via SNMP (Simple Network Management Protocol) traps, e-mail and relay control). However, one disadvantage of this solution is the additional cost of procuring this hardware.

At the time of writing, compatibility tests with the device server and the operating system (OS) hosting the containerised producer application are ongoing.

2. Solution using a Raspberry Pi:

The setup for the Raspberry Pi solution is similar to the set up proposed in Figure 17. The difference is that the dedicated device server is exchanged by a Raspberry Pi computer. Hence, this solution approach can be seen as more cost-effective solution with lower costs but higher integration complexity.

The idea is that the Raspberry Pi serves as a USB/Ethernet data converter that bridges a connection between the depth camera via the USB 3.0 interface and the Edge Cloud system via the Ethernet interface. This approach follows a client-server architecture, where the Raspberry Pi is the server and the Edge Cloud system is the client. However, this approach requires the implementation of a sophisticated communication mechanism in order to minimise the additional latency caused by the Raspberry Pi, given the relatively weak computing power of the Raspberry Pi.

To initialize a connection between the client and server, both need to detect each other. To do so the server listens on a “start” signal. To start the acquisition of data using the depth camera, the client sends a “start” signal to the IP address of the server. Upon receiving the “start” signal a TCP connection between client and server is established, RGB and Depth frames are captured and synchronized using the implemented capture pipeline on the server. Subsequently, the frames are resized and sent in smaller chunks to the client in order to comply with TCP transmission requirements.

At the time of writing, a feasibility check of this option is ongoing, since there might still be some conditions where the Raspberry Pi server is running but not in a state to transmit.

3.3.4 Additional Components

As seen in Figure 16, the Edge Cloud system is linked to both the controlled 5G network environment and the Internet. In scenarios where peers are situated in distinct networks, Internet access becomes imperative for initiating the WebRTC connection.

In this particular use case, the signalling, and Session Traversal Utilities for NAT (STUN) servers are located outside Deutsche Telekom's 5G test network. Instead, they reside in an Ericsson network that is openly accessible to the peer devices participating in the use case. Nevertheless, upon request, there is the possibility of exploring the deployment of signalling and STUN instances within the 5G network testbed of Deutsche Telekom to mitigate delays in the connection set up procedure.

3.3.5 Validation

The integration process is still ongoing. The performance evaluation will be done in the next project phase.

3.3.6 Summary

To summarize, the application platform consists of a communications and processing part.

The components of the communication parts such as STUN and signalling servers are hosted at the Ericsson network with the possibility to deploy the instances to a common network, where the peers of the holographic human-to-human **communications** use case are located.

The components of the processing parts, namely the producer application and the consumer applications are directly implemented into the network testbed.

To integrate the producer application, it is containerized and deployed on the Edge Cloud system. This demands the use of additional hardware to connect the depth camera on the producer side to the Edge Cloud system hosting the producer application responsible for processing and streaming the acquired data by the depth camera. The additional hardware mainly serves the purpose to relay USB data to an Ethernet network where the Edge Cloud resides.

The consumer application integration demands the use of a mobile phone that can be connected to the network testbed using a provided Deutsche Telekom SIM card. In addition, the mobile phone hosting the consumer application must be supported by the corresponding AR glasses of this use case.

The ongoing investigations are centred on integrating the producer application, with a specific focus on the connectivity between the depth camera and the Edge Cloud system. The consumer application has been successfully integrated.

Table 1 highlights important specifications of the implemented use case:

TABLE 1: USE CASE: HOLOGRAPHIC COMMUNICATION SPECIFICATIONS

Holographic Communications	
Users	<p>Producer: a user who is captured by a depth camera. The captured information is used to generate a digital 3D representation which is streamed to a receiving consumer.</p> <p>Consumer: a user who receives, processes, and displays the human 3D representation of the user on producer side.</p>
Network	RTC P2P (WebRTC)
Network Access	<p>Wireless, e.g., Wi-Fi / 5G</p> <p>Wired, e.g., Ethernet</p>
Components	<p>Depth Camera (Intel RealSense): capture of media and spatial information</p> <p>Device Server/ Raspberry Pi: relay instance between camera and cloud.</p> <p>Edge Cloud System (Windows/Linux): generation of meshes for each RGB-D frame and transmission of the data.</p> <p>Mobile phone (Android): receiving, decoding, and rendering of data.</p> <p>AR glasses (Xreal Light): displaying of 3D object to user.</p>

The use case includes two users engaged in digital **communications**. The user on producer side referred provides data to be processed and ultimately transmitted to a receiver on consumer side for the presentation of volumetric content, manifested as a human hologram on AR glasses.

For data exchange, users establish a real-time communication peer-to-peer (P2P) connection employing WebRTC, as described in D3.1 [2].

At the time of writing the application platform supporting the use case is integrated into the network testbed, thereby furnishing the requisite network access technology, whether wireless or wired, contingent on the specific scenario under consideration.

Integration with the network testbed offered by the 5GIC at UoS is currently investigated to complement the SPIRIT platform. This potential interconnection is intended to facilitate the support of heterogeneous network communication scenarios as part of the SPIRIT project. Additionally, it is planned to integrate one-to-many and many-to-many conferencing as future extensions, as described in D3.1 [2].

The essential devices required for leveraging the presented application platform can be supplied by the respective application platform provider, namely Ericsson. This provision aims to facilitate smooth development, integration, and testing of innovations by third parties during the first Open Call. The use of other components by third parties is possible but needs to be considered in collaboration with Ericsson and Deutsche Telekom.

3.4 REAL-TIME ANIMATION AND STREAMING OF REALISTIC AVATARS

The integration of the “Real-Time Animation and Streaming of Realistic Avatars” application platform in the testbed provided by T-Systems (as described in Section 2.2) has been carried out in a sequence of steps, from the definition of the requirements to the evaluation and assessment of the performance. A detailed description of the proposed scenario has been described in other deliverables of the project, such as D2.1 [1], where the general architecture of the framework is included or D3.1 [2], where a list of the innovations developed within the use case is detailed.

3.4.1 Common Prerequisites for Integration

The requirements of both the application platform and the testbed had to be put in common in the first place. The testbed allows the deployment and execution of applications in a container-based environment. This environment offers connectivity options among different containers and access to physical resources, such as GPU cards and computational power. The orchestration of the containers is carried out by a Kubernetes Cluster that uses the Rancher management platform. On the other hand, a 5G internal network is also available to connect the servers with different client devices, such as mobile phones, tablets, and AR glasses. No external connection is needed at this step of the integration since the whole application platform can be tested internally.

3.4.2 Containerization of the Application

The original server application platform was based on a central Unity app in which different external plugins provided the required functionalities. In order to integrate it in the testbed, an effort has been carried out to build a set of two containers from the application. These containers are self-sufficient and include integral environments in which software can run without the need of additional components. This makes this kind of deployments very portable across a variety of scenarios.

Two containers have been created:

- **Audio animation server:** audio processing application written in Python that accepts a set of audio samples and generates a list of the visemes that corresponds to the speech contained in them.
- **Avatar animation, rendering and streaming:** Unity application that acts as a central hub to which the rest of the components interact to generate the mesh and texture of the avatar, render it and stream the resulting audio and video to the clients. This application must run in “server” or “headless” mode. This means that no display can be used on the server. To overcome this obstacle, a virtual display is configured within the container to allow Unity to render the necessary views.

3.4.3 Deployment and Interconnection of Containers

Once the containers were created, they were deployed in the Rancher Kubernetes orchestrator. Two images, corresponding to both necessary containers, were created and uploaded to Docker Hub. Then, a Deployment element was created in Rancher. This deployment has two containers, each one of them using one of the images.

The audio animation server exposes the port 6666 through the container. The main container uses this port to insert the received audio samples so they can be processed. This container, on its side, is exposing the port 8080. This allows the client devices to connect to the running server.

A schema of the architecture of the whole integration can be seen in Figure 18.

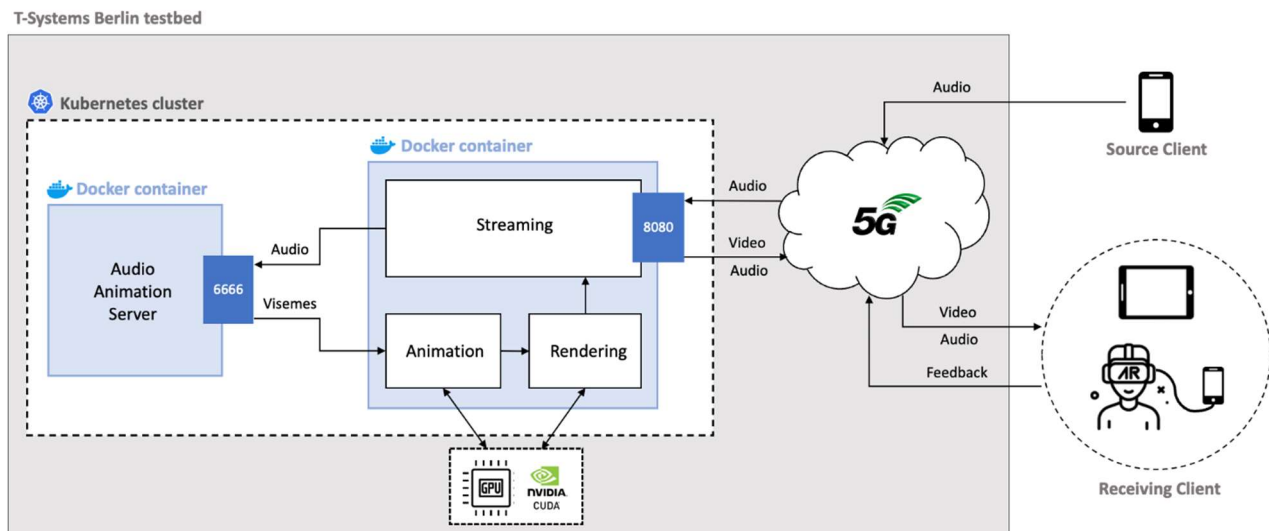


FIGURE 18: INTEGRATION OF THE REAL-TIME ANIMATION AND STREAMING SYSTEM IN THE T-SYSTEMS TESTBED

One of the main challenges that the deployment of the server component has presented is the need of the central Unity application (see D2.1 [1]) to have access to a display while running in headless mode. In this working mode, there is no visible rendering of the Unity scene at the server since the whole rendering process takes place on the background and only the clients show the corresponding video. To solve this, a reference to an X virtual server has been included within the Docker image. This way, the container can access the GPU and the Unity application can run without a real display.

3.4.4 Compatibility Test of Client Devices

A set of testing devices were available to test the connectivity of the 5G network present in the testbed and the overall performance of the scenario. This set included mobile phones such as Samsung Galaxy S20, Samsung Galaxy S8, Oppo Find X2 Pro, Oppo Find X3 and Motorola Edge +.

In a first attempt, important difficulties were met when trying to use the 5G network, since none of these devices were able to connect. This presents a serious challenge, since only specific phones are compatible with the AR glasses that are being used (Xreal Light). Only one phone model belonging to Ericsson, the Oppo Find X2, could successfully connect to the network using a 5G SIM card provided by T-Systems, as specified in Section 3.3.2.

3.4.5 Validation and Performance Evaluation

The performance evaluation will be done in the next project phase.

3.4.6 Summary

The deployment of the application corresponding to the Real-Time Animation and Streaming of Realistic Avatars use case is contained completely within the infrastructure of the available testbeds. A list of the agents that participate in the system as well as the network components used at the testbed can be found in Table 2.

TABLE 2: USE CASE: REAL-TIME ANIMATION AND STREAMING OF REALISTIC AVATARS

Real-Time Animation and Streaming of Realistic Avatars	
Users	<p>Producer: a user whose voice is captured by the media capture device and streamed over the network to the server, where it will be processed and used as input to animate the avatar.</p> <p>Consumer: a user that receives audio and video from the server that is displayed on the client device. This user can interact with the avatar and modify its position, scale and rotation.</p>
Network	RTC P2P (WebRTC)
Network Access	Wireless, e.g., Wi-Fi / 5G
Components	<p>Edge Cloud System (Linux 20.04 LTS): generation of the animated avatar, rendering, streaming, Websockets/WebRTC connection management.</p> <p>Media capture device: Android mobile device that captures media (audio) and streams it to the server through the network.</p> <p>Consumer device: Android mobile device that displays audio and video received from the server and can interact with the avatar.</p> <p>AR glasses (Xreal Light): Augmented Reality glasses that displays audio and video received from the server and can interact with the avatar.</p>

The extensive compatibility of the system with a variety of devices, especially Android mobile phones or tablets, reduces the number of obstacles encountered when trying to connect a media capture/consumer device. On the other hand, only a few mobile phones are compatible with the investigated AR glasses, so a potential user must convey with this information when using the application. The current limitations of the 5G network available in the Berlin testbed must also be taken into account.

4 IMPLEMENTED USE CASES

This section covers in detail the implementation of use cases, which are described in detail in [1]. On the one hand, the integration of the platform components into the use cases and the description of the necessary adjustments for a smooth integration into the testbeds and, on the other hand, a look ahead to how these components might find use in various scenarios, such as third-party applications. For each use case, there is a table at the end of the respective section that summarises the relationship to the requirements and KPIs from the document [1], together with the validations carried out so far.

4.1 UC HOLOGRAM: HOLOGRAPHIC HUMAN-TO-HUMAN COMMUNICATIONS

This section presents the application pipeline and possible extensions of the holographic human-to-human communication use case implemented into the network testbed representing one part of the first version of the SPIRIT platform. The use case can be extended within the SPIRIT project to offer a wide compatibility with applications from third parties.

4.1.1 Overview

Figure 19 presents the implemented holographic human-to-human use case using the application platform developed by Ericsson and network testbed provided by Deutsche Telekom:

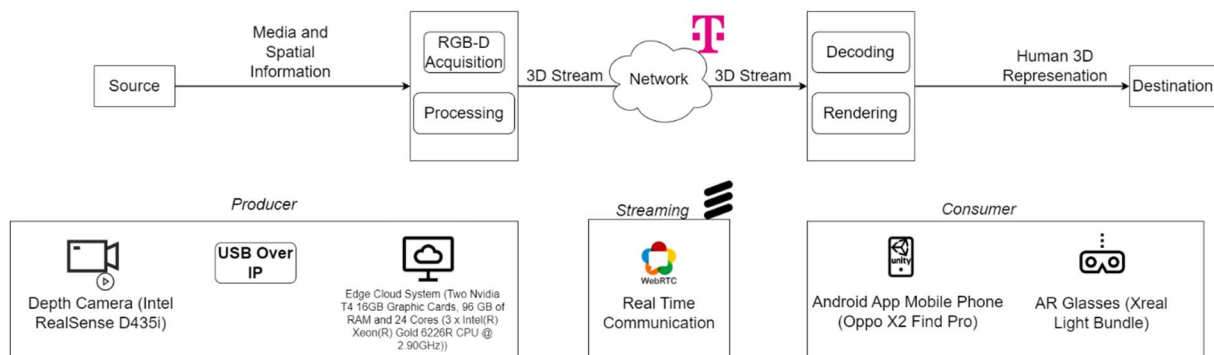


FIGURE 19: APPLICATION PIPELINE OF THE HOLOGRAPHIC HUMAN-TO-HUMAN COMMUNICATION

The use case is implemented by means of two applications, i.e., a producer and a consumer application:

- **Producer application:** Containerised Python-based application hosted on a Windows/Linux OS connected to a depth camera utilising a dedicated device sever or configured Raspberry Pi.
 1. Upon starting the container, Edge Cloud connects to signalling server hosted at Ericsson.
 2. Once the consumer device joins the signalling server, a WebRTC channel is set up as described in D3.1 [2].
 3. Producer application initiates the acquisition of RGB-D information data of an object of interest (human torso/face) using a depth camera.

4. Producer application generates human 3D representations using the acquired RGB-D information.
 5. Encoded human 3D representations are streamed to the consumer application using wireless or wired network access technologies provided by Deutsche Telekom network testbed.
- ➔ **Consumer application:** Unity (C#)-based application running on an Android mobile phone connected to AR glasses via USB-C.
1. Upon starting the consumer application, the mobile phone connects to signalling server hosted at Ericsson.
 2. Once, the producer device joins the signalling sever, a WebRTC channel is set up as described in D3.1 [2].
 3. Consumer application receives the 3D data stream application with the help of wireless or wired network access technologies provided by the Deutsche Telekom network testbed.
 4. 3D data stream is received, decoded, and rendered.
 5. Human 3D representation of user on producer side is displayed on AR glasses in real-time.

4.1.2 Extension of Use Case with Third Party Applications

The implemented use case offers real-time holographic streaming delivery with the flexibility of manual configuration for both quality and data size of holographic content. This includes the ability to achieve a 3D representation through filtering and compression techniques that can be tailored to meet network and Quality of Experience (QoE) requirements. Here, a producer client application functions as a versatile tool for media and spatial information acquisition and processing, compatible with applications supporting either network or USB connections. On the other end, the consumer client application serves as a decoding and rendering component. It allows users to visualize encoded 3D data transmitted from the producer client application. Both application parts are adaptable to wireless network access technologies such as Wi-Fi or 5G, providing a seamless and efficient holographic communication experience.

For the SPIRIT project the goal is to extend this implemented use case by leveraging immersive telepresence application provided by third parties in the scope of two Open Call waves.

General

- **Research on 3D data processing**
External parties are given the opportunity to utilise the use case to drive development in the processing of 3D content. This includes researching and testing novel processing methods in the areas of point cloud generation and mesh calculation as well as encoding/compression algorithms. Assessments by third parties can be made on various aspects such as encoded data size, processing time, image quality, or overall QoE.
- **Implementation of various streaming technologies**
The existing iteration of the application employs WebRTC for streaming 3D data between the clients asymmetrically. There are ongoing developments in extending these

technologies, e.g., in terms of scalability. This offers a promising chance to explore the system's performance. In addition, alternative streaming frameworks such as LL-DASH may be explored and compared to WebRTC solutions.

- **Use of new devices and new ways of interaction**

The market of AR devices evolves rapidly and new options with improved features are constantly made available. This presents a good chance to extend the compatibility of the application to a wider range of equipment. New ways of interaction between the user and the holographic content could be implemented using new device technologies to provide a richer immersive experience. This for example may include the incorporation of the Microsoft HoloLens 2 or the Apple Vision Pro through the third parties.

Furthermore, in the first version of the SPIRIT platform, the following extensions are conceivable but should not be understood as limiting the immersive telepresence application of third parties:

Producer Application

- *Incorporating AI:* Real-time capturing systems, as present in the implemented use case, encounter several challenges that impact the overall quality of the captured content. These challenges include operating within strict time constraints, often requiring swift and efficient processing. Additionally, issues related to low-resolution textures can affect the clarity and detail of the captured images. Bad lighting conditions during the capturing process pose another hurdle, impacting the overall visual quality. Moreover, artifacts in both geometry and texture, such as holes and noise, present further complications. These irregularities can detract from the accuracy and fidelity of the captured human 3D representation.

Third parties could address real-time system challenges through research and development activities. In addressing these challenges, a comprehensive approach is necessary to ensure optimal performance and quality in real-time capturing scenarios.

For example, one approach could include the development, implementation, and validation of machine learning solutions for this use case similar as done in [4]. The authors make use of a novel technique called “Neural Re-Rendering”. Here, a neural network architecture is utilised to perform operations such as completion, super resolution and denoising on a captured hologram in real-time, as depicted in Figure 20:

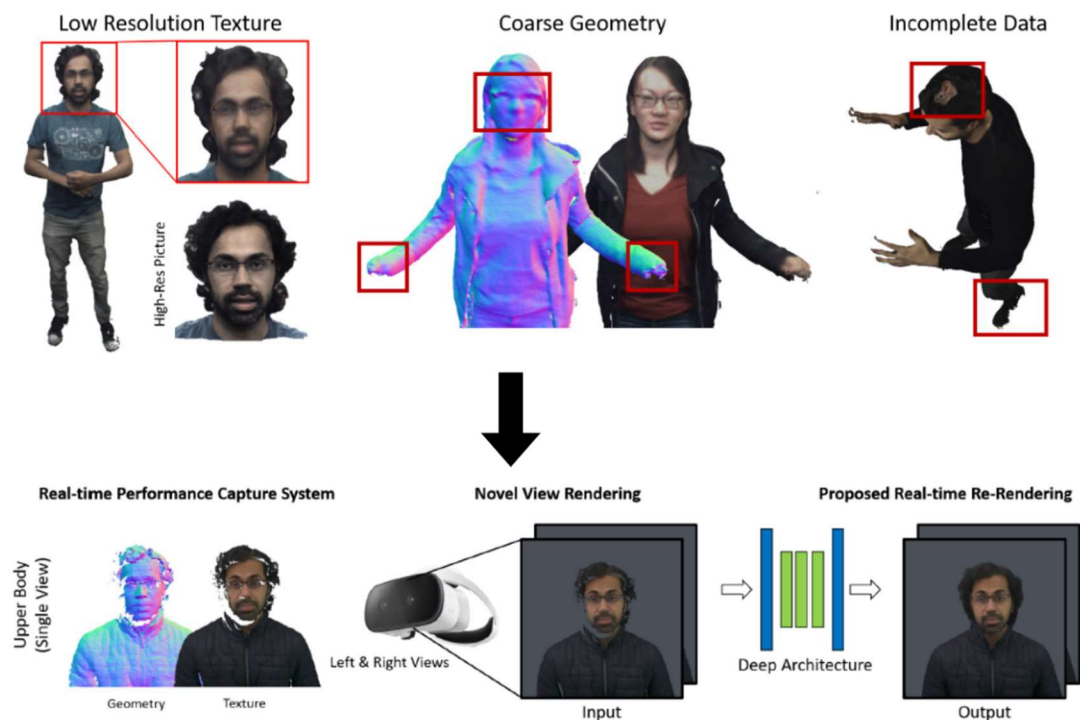


FIGURE 20: NEURAL RE-RENDERING FOR REAL-TIME PERFORMANCE CAPTURE SYSTEMS [4]

- *Incorporating Apple iPhone:* Another extension of the use case with third parties may be the exchange of the depth camera with an iPhone 14 or 15 since those models offer LiDAR sensors and therefore can replace the depth camera as source to capture RGB and depth information that can further be processed by the producer application.

Consumer Application

- *Incorporating edge computing (split-rendering):* The rendering operation is done by the consumer application supposed to be running on a mobile end device such as a commercial off-the-shelf mobile phone. However, rendering is GPU-intensive, as the processes involved in creating realistic and visually appealing images or animations are very complicated. The complexity arises from factors such as intricate scene geometry, high-resolution textures, advanced lighting effects and complicated shading models. Meticulous calculations for light interactions, reflections, refractions, and shadows are required to achieve photorealistic results as demanded by the digital human-to-human use cases presented in D2.1 [1]. One solution to mitigate the impact of rendering on hardware with low GPU-power is to (partially) off-load the rendering operation on a more sophisticated hardware, such as the Edge Cloud system provided in the implemented use case.

Similar to [5], third parties could investigate cloud solutions leveraging the provided Edge Cloud system for the use case with a special focus on rendering within immersive telepresence scenarios. Excluding cloud service monetization, computational off-loading of heavy software operations from a mobile end device to a cloud system not only optimises end consumer battery usage but also contributes to possibly reducing end-to-end latency in the overall system originating from faster software processing time.

4.1.3 Summary Validation

TABLE 3: USE CASE HOLOGRAMS: DEVELOPMENT/INTEGRATION AND PERFORMANCE

Use case Avatar development and integration	Relevant requirements	KPIs (metrics achieved)	Evaluation (determination of KPIs)
<p>The use-case was developed by EDD and initially tested locally at 5G testbed in Aachen.</p> <p>Integration in the Berlin DT testbed is ongoing.</p>	<p>RLat</p> <p>RDown</p> <p>RClients</p> <p>RSignal</p>	<p>A qualitative evaluation will be provided in the next revision.</p> <p>Clients = 2</p> <p>Signal = WebRTC signalling server</p>	<p>E2E latency (Lat) is in this case the time between the capturing and displaying on AR glasses.</p> <p>Downlink bandwidth corresponds to the 3D video stream.</p> <p>Two clients were used in the tests, one producer and one receiver.</p> <p>A signalling server included in the application was used to test the communication.</p>

4.2 UC MULTI-SOURCE: LIVE TELEPORTATION WITH 5G MEC

4.2.1 Overview

This use case is about live teleporting people from remote internet locations to a common virtual space of the audience such that the audience can have the immersive and multisensory perception that everyone is located in the common physical scene. One application scenario is distributed virtual performances where actors can physically perform (e.g. dancing) at different locations but their live holograms can be simultaneously teleported to a “virtual stage” where the audience can enjoy the entire performance event constituting the virtual holograms of real performers from different remote locations.

4.2.2 Use of the System within the Surrey Testbed

The overall application platform has been implemented based on open-source platform of LiveScan3D [38]. At content source side (e.g. person to be captured and teleported), multiple Kinect Azure DK cameras are used to capture the object from different directions and each camera is connected to a local PC called clients which is responsible for local processing of the raw content data. The raw content data is in point cloud data format.

At the 5G Multi-access Edge computing (MEC) server side, the pre-processed local data are further streamed to the production server responsible for integrating frames produced from different clients for the same captured object. The functionality of the production server is supported by a 5G MEC attached to the 5G testbed network at University of Surrey. In this case local frames from individual clients are streamed in real-time to the local 5G MEC through 5G new radio (NR) uplink.

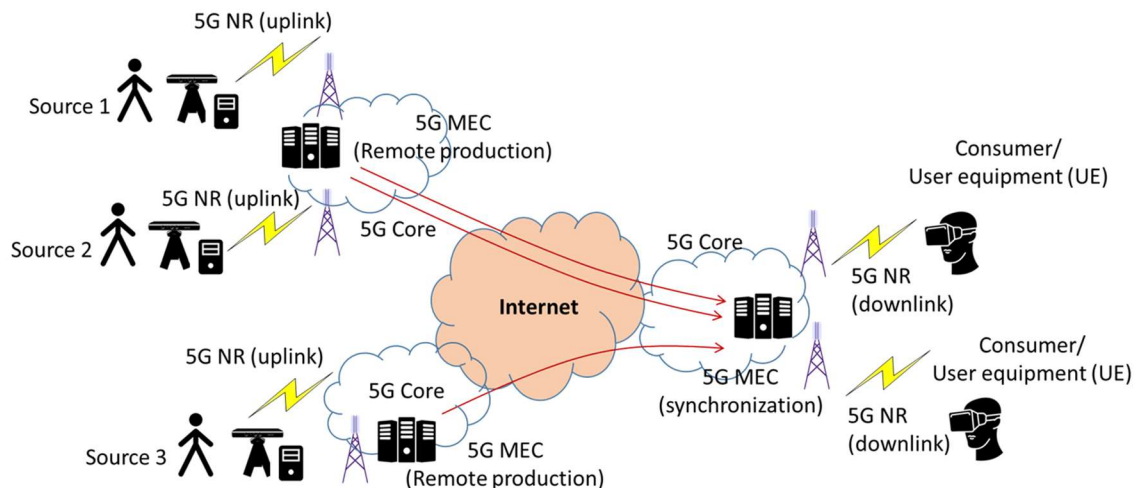


FIGURE 21: IMPLEMENTED FRAMEWORK FOR LIVE TELEPORTATIO

The use of this platform can be described as follows:

- 1) **Start LiveScan3D server at MEC server:** on the MEC server, first the LiveScan3D server should be started. Then at the control panel, calibration parameters including offset for each source can be tuned as well as point density, request interval, buffering control parameters.
- 2) **Start LiveScan3D clients at different sites:** once one or multiple Microsoft Azure Kinect DK cameras can be deployed on different sites and connected to local laptop, LiveScan3D client can be launched at each laptop. Local resolution level and depth mode can be selected on the application panel and local source ID should be configured. Especially, the source ID should be predefined to avoid collision with other clients.
- 3) **Connect clients to MEC server and streaming Live teleportation:** The client can set the remote MEC server IP as the server IP to be connected, and once the connect button is clicked, a TCP connection will be requested and then established from client to MEC server. Then the live hologram can be requested from remote MEC server. On the MEC server panel, by click "show live" button, real-time hologram from multiple sites can be merged and displayed on the screen.
- 4) **Start HoloLens 2 and receive the holographic streaming:** once the live teleportation from clients to server is running, the user can start the HoloLens 2 to connect to the same MEC server via TCP connection to request live teleportation streaming. Each HoloLens 2 will have dedicated user ID, then the server can decide how many sources should be streamed to this specific user. During the streaming, a user can perform several actions to interact with the hologram including moving closer or far away, manipulating the hologram to change its size, orientation.

4.2.3 Summary Validation

TABLE 4: USE CASE MULTI-SOURCE: DEVELOPMENT/INTEGRATION AND PERFORMANCE

Use case Multi-Source development and integration	Relevant requirements	KPIs (metrics achieved)	Evaluation (determination of KPIs)
<p>The use case was developed by University of Surrey and integrated into Surrey’s testbed. The current UC status is tested under the Surrey’s 5G network testbed.</p> <p>Each source consists of a unique client which encompasses a Microsoft Azure Kinect Dk/Kinect-2, and corresponding computing resource, utilized by the source client application for the generation and transmission of the relevant point cloud data to the server.</p> <p>The primary viewer client device utilized at the receiver is the Hololens 2.</p> <p>Major challenges in integrating the UC were in relation to computing resources and network configurations. This indicates that performances could be impacted to some degree by such factors. Also, integration and development are not inclusive of the WebRTC protocol, with the subsequent inclusion being a planned and ongoing actionable item.</p>	<p>RLat</p> <p>RDown</p> <p>RUp</p> <p>RClients</p> <p>RQoE</p> <p>RFPS</p> <p>RRes</p>	<p>Lat = 300 ms</p> <p>Down = 50-70Mbps</p> <p>UP = 50-70Mbps</p> <p>Clients = 3</p> <p>QoE = ~3-3.7</p> <p>Frame synchronization window = 100ms</p> <p>FPS=25-30</p> <p>Res= ≤ 1280 x720(512x288)</p>	<p>RLat is determined by the processing capability of the server running LiveScan3D platform as well as the network conditions (e.g. network delay). This can be up to 300ms in challenging network conditions.</p> <p>The 3 clients used in the tests represent 2 unique producers/sources and a receiver.</p> <p>The QoE metric incorporates subjective driven model-based QoE evaluation for volumetric media.</p> <p>Downlink and uplink bandwidth (Down and Up) in the 5G network of up to 70Mbps respectively, as measured in separate experiments.</p> <p>Regarding synchronization performance, this also depends on the network distance from individual sources to the viewer side.</p> <p>FPS is within the 25-30 range and corresponds to the resolution level at ≤ 1280 x720(512x288) resolution which is indicative of the relative point density. Additional Tests with other resolutions are planned.</p>

4.3 UC AVATAR: REAL-TIME ANIMATION AND STREAMING OF REALISTIC AVATARS

4.3.1 Overview

The Real-Time Avatar Animation and Streaming application proposes a communication scenario between a producer user, whose voice is captured and used as input to animate a volumetric avatar, and a consumer user, that receives video and audio information of the animated



avatar and has the possibility of interacting with it. Once again, the reader is invited to refer to the documents D2.1 [1] and D3.1 [2] for a more detailed explanation.

This section describes the steps needed to run and test the application platform within the testbed provided by T-Systems in Berlin, as well as a list of possible extensions of the application that could be carried out by third parties.

4.3.2 Use of the System within the Berlin Testbed

A step-by-step description of how the application can be used within the Berlin testbed is described here. At the time of writing, and as mentioned in previous sections, there are still several issues that prevent the system from being fully integrated in the infrastructure provided by T-Systems.

The use of the Real-Time Animation and Streaming of Realistic Avatars application involves the following steps:

1. **Start audio animation server:** the container associated to the audio animation server must be running before starting the main container. This server receives a stream of audio samples and generate the visemes (speech sound descriptions) used to animate the avatar in a further step.
2. **Start rendering and streaming server:** the main container must be started by selecting the necessary parameters. These include, among other options, the working mode (single image or stereoscopic), and the desired resolution of the rendered image.
3. **Connect media capture device to server:** the producer user must start the media capture Android application, enter the IP of the server within the network and connect using Websockets.
4. **Connect consumer device to server:** the consumer device can consist of a single Android mobile phone or tablet or a device connected to a set of AR glasses. In both cases, the user can interact with the avatar.
 - a. Single Android device: the user must start the corresponding Unity application, enter the IP of the server and connect using Websockets. The image of the avatar will appear directly on the screen and the audio will be played on the device.
 - b. AR glasses: the user must connect the glasses to the corresponding Android phone, start the Unity application, enter the IP of the server and connect to it. The image of the avatar will be displayed on the glasses and the audio will also be played on them.

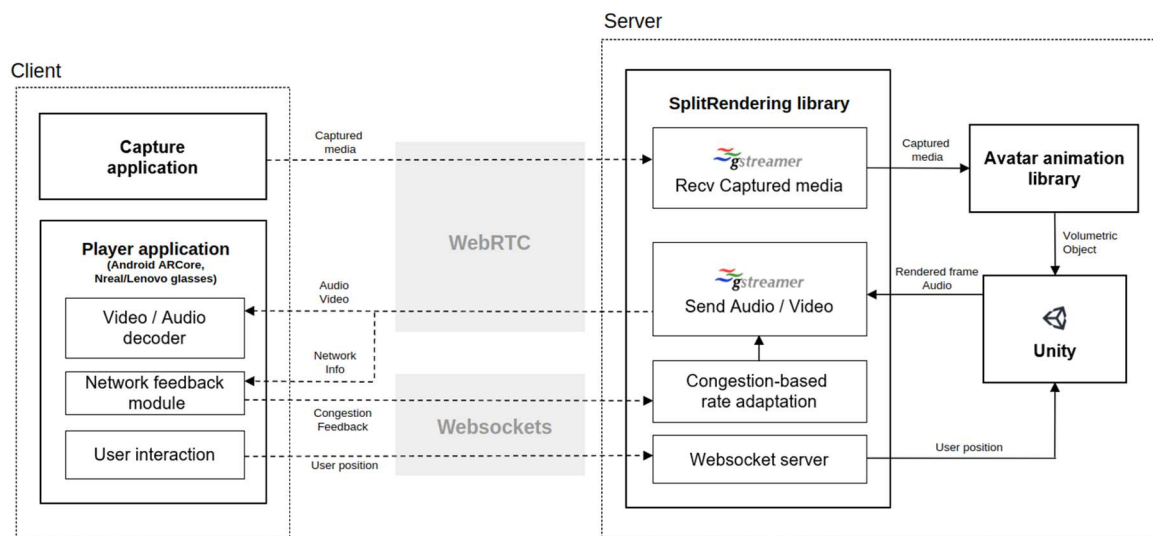


FIGURE 22 COMPONENTS OF THE REAL-TIME ANIMATION AND STREAMING OF REALISTIC AVATARS APPLICATION

On the other hand, the flow of data of the whole scenario can be described in the following steps (as described in Figure 22):

1. Audio is captured on the producer device.
2. The audio samples are streamed over the network to the server using a WebRTC connection.
3. A set of audio samples (corresponding to the current frame) is used as input for the avatar animation library.
4. The avatar animation library sends the audio samples to the audio animation server and visemes are generated.
5. The produced visemes are used to generate the mesh and texture of the avatar for the current frame.
6. The mesh and texture of the avatar are updated within the Unity scene.
7. The position of the client device with respect to the avatar is synchronized on the server and the cameras in the Unity scene are placed at the right position.
8. The transformations resulting of the interaction of the consumer user with the avatar (position, rotation, scale) are applied at the server.
9. One or two 2D images (depending on the working mode) are rendered by the corresponding cameras.
10. The 2D images are streamed over the network to the consumer device, together with the audio corresponding to the current frame.
11. Both audio and video are displayed on the consumer device.

4.3.3 Extension of Use Case with Third Party Applications

The architecture of the application makes it possible to extend its functionalities in a variety of ways within the development of the SPIRIT project. Each of the main components (server application, producer client and consumer client) provide a series of subsystems that can be

further developed to improve the already-present functionalities or to add new ones. This section includes some entry points for third parties to do this.

Use of different avatars or animation methods

The avatar included in the first version of the application platform can be switched to a new one with different characteristics. The architecture of the system allows any object placed in a Unity scene to be rendered and shown on the client producer device. This opens up a number of possibilities regarding the aspect of the avatar and the way it is animated frame by frame. New animation algorithms could be used as libraries within the existing infrastructure.

Manipulation of the rendered images and audio

The nature of the rendering procedure used in this application makes it possible to manipulate the generated 2D image in an infinite number of ways. New techniques based on the use of AI (Artificial Intelligence) and Neural Networks can be used to improve the quality of the renders or to modify its aspect.

The same can be applied to the audio captured on the producer client device. The samples can be processed in different ways, providing better sound quality, noise removal or voice manipulation.

Use of different streaming technologies

The current version of the application uses Gstreamer and WebRTC to stream audio and video information from the server to the client and vice versa. New extensions of these technologies are being developed at the moment. This presents a good opportunity to investigate how the system behaves when being integrated with other streaming frameworks.

Use of new client devices and new ways of interaction

The market of AR devices evolves rapidly and new options with improved features are constantly made available. This presents a good chance to extend the compatibility of the application to a wider range of equipment. New ways of interaction between the user and the avatar could be implemented as well in order to provide a richer experience.

Measure of Quality of Experience and Performance

The existing capabilities of the system, together with the possible new additions described above, provide an interesting playfield to analyse and compare the behaviour of different rendering and streaming technologies in the context of the communication in Mixed Reality environments.

4.3.4 Summary Validation

TABLE 5: USE CASE AVATAR: DEVELOPMENT/INTEGRATION AND PERFORMANCE

Use case Avatar development and integration	Relevant requirements	KPIs (metrics achieved)	Evaluation (determination of KPIs)
<p>This use case was developed by Fraunhofer HHI and integrated in the Berlin DT server in collaboration with T-Systems.</p> <p>The necessary applications were containerized and deployed in the Kubernetes cluster of the testbed. Then, the main functionality (animation and streaming of the avatar) was tested.</p> <p>The main difficulties found during the integration process were related to the compatibility of the client devices with the 5G network. One of the Android smartphones (Oppo Find X3) could finally be connected and used for the tests.</p>	<p>RLat</p> <p>RDown</p> <p>RClients</p> <p>RSignal</p>	<p>Lat = ~300 ms</p> <p>Down = 5 Mbps</p> <p>Clients = 2</p> <p>Signal = WebRTC signaling server</p>	<p>E2E latency (Lat) is, in this case, the time between the moment in which the producer user sends audio from the phone and the instant that the receiving user hears the audio. Some optimizations must still be done to achieve the goal E2E latency.</p> <p>Downlink bandwidth corresponds to the video and audio information. The mono modo was tested and the bandwidth was as expected.</p> <p>Two clients were used in the tests, one producer and one receiver.</p> <p>A signaling server included in the application was used to test the communication. Since the application was tested locally, no external connections with STUN/TURNS server have been used yet.</p>

4.4 REAL-TIME HUMAN-MACHINE INTERACTIONS

4.4.1 UC Robot: Distributed Steering of Autonomous Mobile Robots (AMRs)

Alongside the SPiRiT Platform the Deutsche Telekom Testbed provides a teleoperation use case that is available on premise. The use case is about the remote manual navigation of autonomous mobile robots and is further described in the D2.1 document [1].

The use case runs centralized on a local edge server and can be used by any device (for example phone, tablet, controller and computer) with a web browser (preferably chrome) inside the campus. Within the web application one can view the robot’s surroundings by observing video streams from the AMR as well as sending out drive commands to the robot. In general, the web application is capable of manually and autonomously steering multiple different AMR’s however only one robot was made available for the project, the “Husky” Robot Figure 23: Husky Robot from ClearPath Robotics.





FIGURE 23: HUSKY ROBOT

In order to enable the robot for autonomous driving and distributed steering, it had to be equipped with multiple different sensors. We use IMU sensors to detect acceleration and pose, use Lidar sensors to perceive the robot's environment and cameras to provide assistance for remote workers. The Lidar sensor that is being used on the Husky is the 3D Ouster Sensor OS0. For the cameras we are using four Intel depth cameras d435, however they are implemented as normal RGB cameras for now.

To communicate with our fleet management system and distributed steering use case that is running on the edge cloud, the Husky is connected via a 5G modem.

For the presentation of the use case a simplified control center has been setup with six 1920x1080p displays and an Intel Nuc computer which can be seen in Figure 24: Simplified control center.



FIGURE 24: SIMPLIFIED CONTROL CENTER

Side by side with the web application we also offer an API to control the robot to participants of the SPIRIT project. For safety reasons the API can only be used in agreement and with the guidance of the Autonomous Logistics Team from the Deutsche Telekom. We have a variety of APIs available however for the interaction with our use case we're providing APIs for the following functions:

- ➡ Sending Drive commands
- ➡ Post/get a map for the navigation of one specific robot

- Defining POI's/Goals in a robot specific map
- Post autonomous driving order for a specific robot to a specific POI/Goal
- Accessing the video stream of Husky robot

However, during the execution of the experiment's special usages and extensions of the API can be discussed.

4.4.2 Summary Validation

TABLE 6: USE CASE ROBOT: DEVELOPMENT/INTEGRATION AND PERFORMANCE

Use case Robot development and integration	Relevant requirements	KPIs (metrics achieved)	Evaluation (determination of KPIs)
<p>This use case was developed by T-Systems and is fully integrated in the Berlin Testbed.</p> <p>A robot that is available to the testbed was equipped with four cameras and a producer client application which forwards video streams to a local edge server. To make use of the local 5G network, the robot was also equipped with a 5G modem.</p> <p>A containerized application was deployed to the edge server providing a web interface for clients to connect to the video streams.</p> <p>For the demonstration of the use case a demo "control center" consisting of an Intel NUC and multiple monitors has been set up.</p>	<p>RLat</p> <p>RDown</p> <p>RUp</p> <p>RFPS</p> <p>RRes</p> <p>RDev</p>	<p>Lat = 200 – 300ms</p> <p>Down = 2 Mbps</p> <p>Up = 2 Mbps</p> <p>FPS = 30 fps</p> <p>Res = 640x480 px</p> <p>Dev = Desktops, Tablets, Smartphones</p>	<p>E2E latency (Lat) was determined with a simple test capturing a stopwatch on the screen of the control center with the cameras of the robot. The resulting latency varied between 200-300ms.</p> <p>Downlink and Uplink were fully satisfied by the local network. The Robot was connected via 5G while the receiver of the video streams was connected via ethernet.</p> <p>The framerate is limited to 30fps by the camera used in the experiment. Tests with other cameras have not been made as of now.</p> <p>The experiment was performed with 4 streams at 640px480p resolution. Higher resolution has not yet been tested.</p> <p>The video streams can be accessed by any device running a web browser. So far android smartphones, iPhones, iPads, windows desktop pcs and linux desktop PCs were tested.</p>



5 SECURITY DEVELOPMENT AND TECHNICAL INTEGRATION

5.1 Problem Statement

In the SPIRIT project, we propose to the implementation and deployment of an open architecture to foster vendor interoperability and ease re-use of our toolbox by partners. This approach leads to specific challenges that must be addressed in our work, specifically from our security viewpoint:

- The different components are disaggregated, i.e. they are functionally and spatially distributed.
- Components might be deployed on cloud or edge devices beyond the control of the users and communication partners.
- The components are provided by different vendors with different run-time requirements.
- Some of the components might run in public cloud environments or edge servers in physically exposed locations that are not under direct control of project partners.

Such a highly dynamic environment creates new attack surfaces and security problems. To this end, we have developed the new approach of end-to-end security (see Figure 25), where data in transit is encrypted and only provably trust-worthy cloud applications can break up this encryption to process and store the confidential data. To exclude the cloud provider itself from the trusted computing base (TCB) of the application, any solution must provide robust security guarantees that can also be verified by remote parties before delivering their potentially confidential data to the cloud.

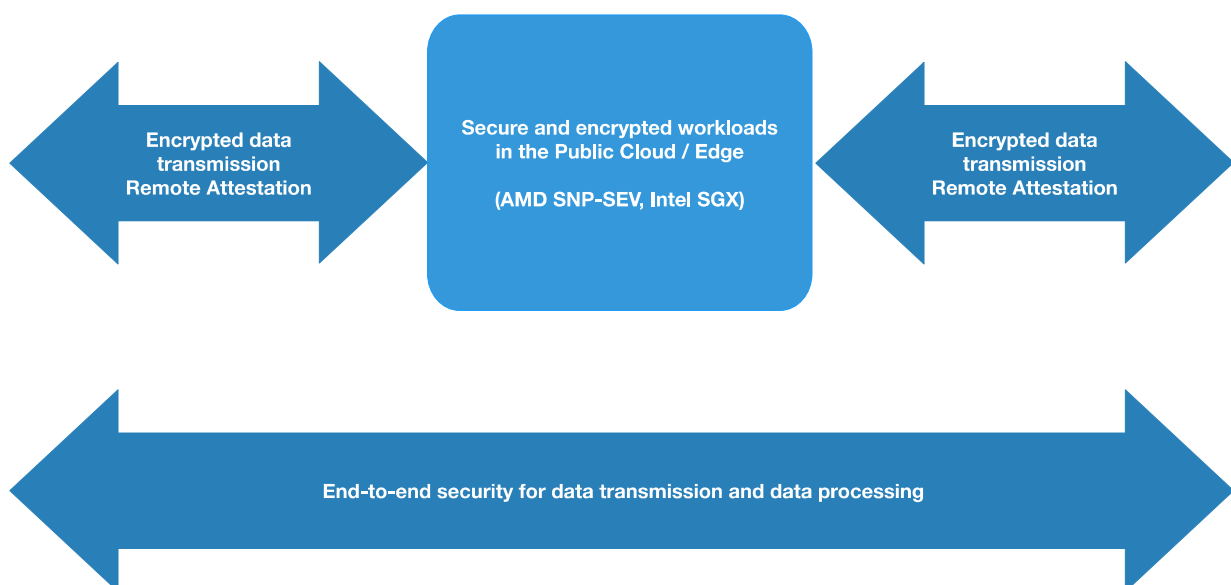


FIGURE 25: END-TO-END SECURITY

In previous reports [2] we have identified cloud-based solutions and their limitations as well as cloud-specific vulnerabilities, such as

- mishandling of customer private keys by cloud providers,
- opaque processes in the cloud, and
- reliance on security certifications instead of hard security guarantees rooted in hardware properties.

In this document we specify an approach for deployment of Confidential Computing-protected virtual machines (VM) on bare metal servers (either on-premise or in the cloud) where private keys for managing VMs never leave the data owner's control and specifically do not need to be uploaded to cloud provider systems. The specification is enhanced with hands-on documentation to help project partner's use the provided software.

Computing resources as well as project partner support is provided by the Deutsche Telekom Cloud Security Lab in Berlin.

5.2 OVERVIEW OF THIS SECTION

This section serves both as a specification of the approach to Confidential Computing we have designed and implemented for the project as well as a user manual for project partners. The section structure supports this dual-purpose approach.

In section 5.3 we introduce our approach to Confidential Computing on a bare-metal Linux system, giving a high-level overview of the processes as well as an application example to further explain the system.

In section 5.4 we specify the different components we have implemented for our solution, notably the tool virtual machine (VM), the REST API for remote management, and the accompanying Docker image to simplify remote management for guest owners. The specification parts are interspersed with some hands-on instructions to support the dual role of user manual of this document.

In the appendix section "A.4 End-to-end Example" everything is put together to give users of our software an end-to-end example of the whole process, from obtaining the necessary software from our download site until the final operation of the trusted VM on our infrastructure.

Finally, in the Appendix A: Security we document specific aspects of our setup that did not fit into the previous.

5.3 Overview of Linux-based Confidential Computing on Bare Metal Server

After a recap of previous deliverables, introducing the trust model and some basic terminology of Confidential Computing, we explain in this section what Confidential Computing facilities are available on a fresh Linux installation (in our case Centos 9 Stream) and what is needed to achieve a practical and secure implementation that addresses the attacker model where the hardware operator of the VM host system is potentially untrustworthy.

5.3.1 Trust Model

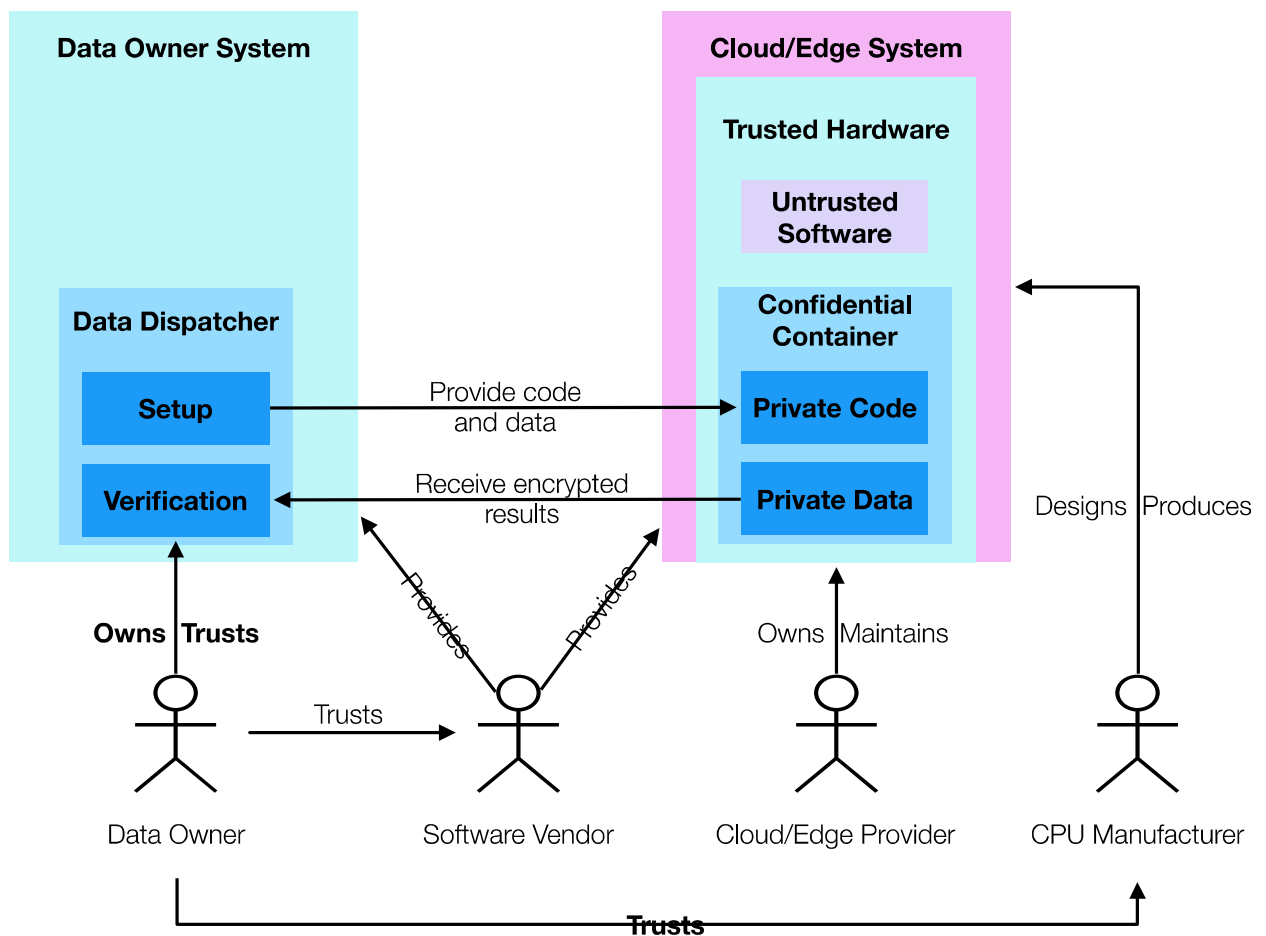


FIGURE 26: TRUST RELATIONSHIP IN CONFIDENTIAL COMPUTING

Confidential or Trusted computing in the context of this document refers to technologies which aim to solve the secure remote computation problem by leveraging trusted hardware in the remote computer. The trusted hardware establishes a secure container, and the remote computation service user uploads the desired computation and data into the secure container. The trusted hardware protects the data’s confidentiality and integrity while the computation is being performed on it.

Figure 26 shows how the users trust the manufacturer of a piece of hardware in the remote computer, and entrust their data to a secure container hosted by the secure hardware.

The Data Owner (also called Guest Owner later on) sets up the remote computation on the cloud or edge server by preparing code and data to be sent to the remote system.

On the remote side, the code and data are run in a Confidential Container (or Confidential VM). The guest or data owner only interacts with the Confidential VM after verifying its integrity through the process of measurement or remote attestation.

5.3.2 Basic Approach to Measured Start of VM

In our host system (representing the cloud/edge system) we run the standard Linux KVM virtualization manager (VM), with the qemu and libvirt software stack. The qemu component allocates resources and starts the open virtualization manager firmware (OVMF, basically an UEFI-based BIOS of the VM), which in turn invokes the grub boot manager which starts the kernel with the initial ram disk image (initramfs) which then mounts the root disk image of the Linux guest (Figure 27).

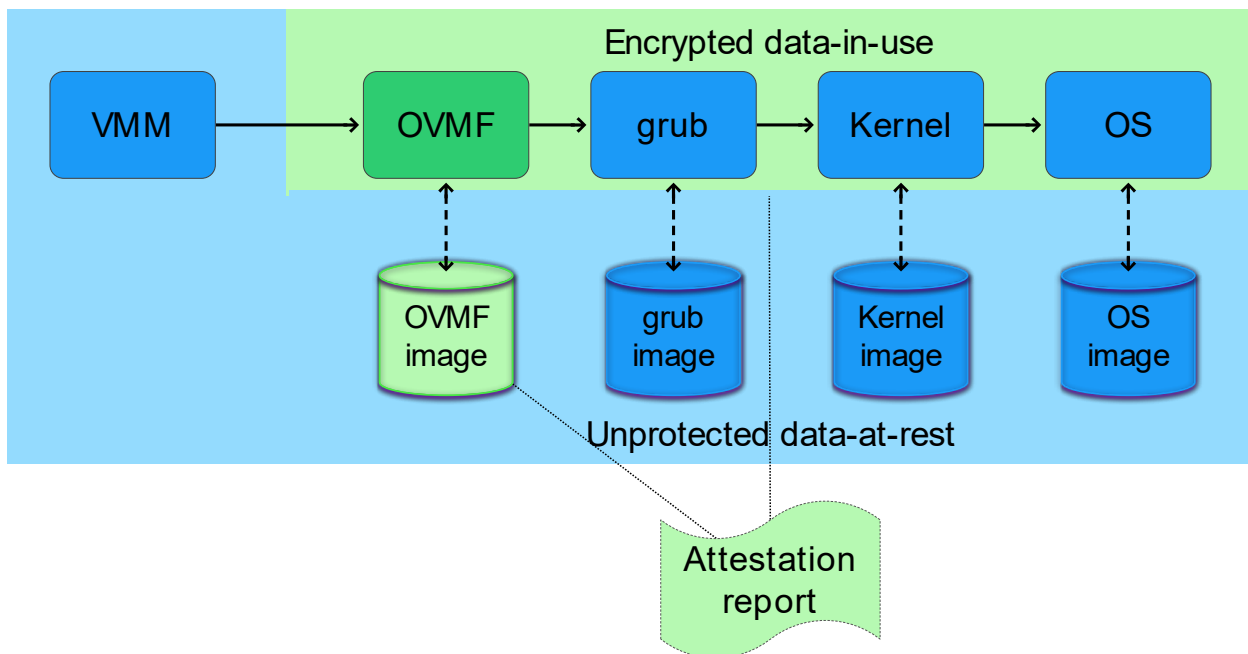


FIGURE 27: BASIC MEASURED START ON LINUX

The challenge in confidential computing is the following: the guest VM operates within a secure envelope, while the host hypervisor (including the QEMU process) operates outside this secure boundary, making it untrusted. Even with the new Luks (*Linux Unified Key Setup*, the Linux standard for disk encryption) format, QEMU handles image encryption, resulting in the encryption key residing outside the secure envelope. As a result, there is a need for a new format that ensures the encryption key (and the encryption mechanism) remains within the secured guest VM to enhance security.

Additionally, the attestation report obtainable with standard host tools only reports the integrity of the host processor, its firmware (in our case AMD Platform Secure Processor firmware), as well as the OVMF firmware image. The grub, Kernel, and OS image reside unprotected on the host system disks and are open to tampering.

5.3.3 Measured Start with Fully Encrypted Disk Images

In order to fully protect the remote environment, grub, Kernel, and OS images must be securely encrypted so that even the cloud provider cannot read or alter data. This, however, leads to the problem of bootstrapping this encryption key in a secure manner. In this section we describe how we achieved that.

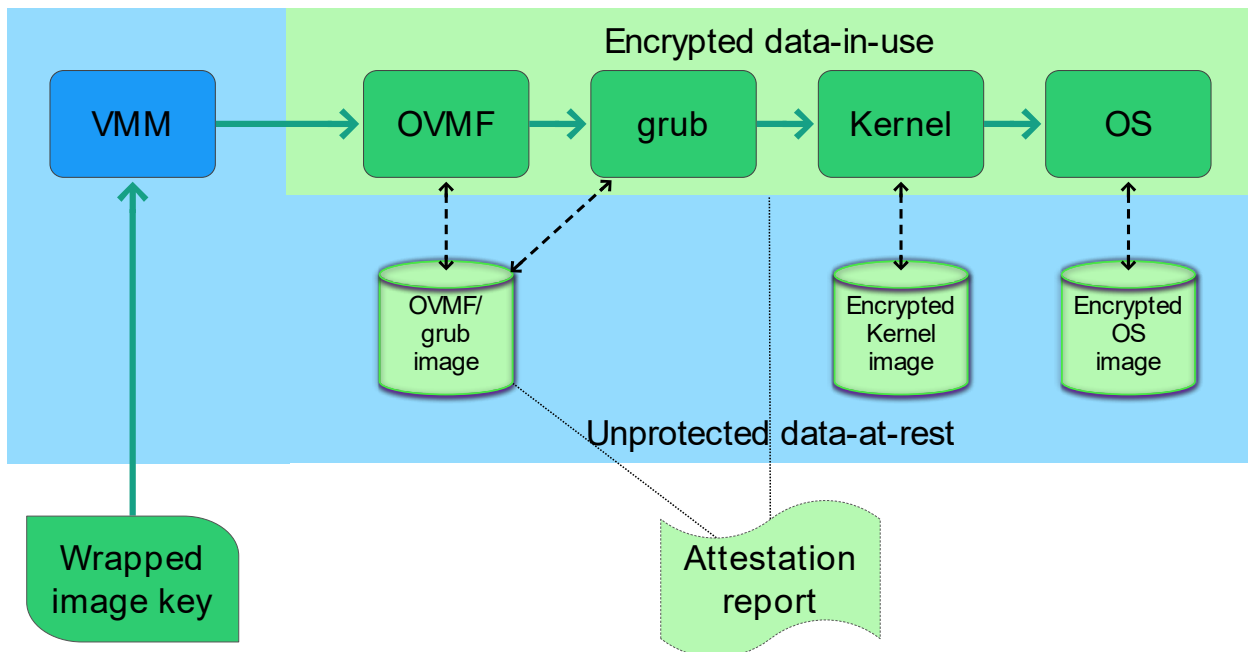


FIGURE 28: MEASURED START WITH FULLY ENCRYPTED DISK IMAGES

Figure 28 shows the boot process after everything is set up:

1. The guest owner instructs the cloud provider to start the remote VM, but first only in paused state. The remote VMM (virtual machine manager – libvirt/qemu) starts the VM with a combined OVMF/grub image.
2. The guest owner requests a measure via the QMP protocol (Qemu Management Protocol). If the measurement meets expectations (OVMF/grub hash, AMD-SEV HW revision, AMD PSP firmware revision), the guest encrypts the Kernel/OS image key for the remote hardware and created a so-called “wrapped image key”.
3. The key is sent via QMP to the remote system and passed on the PSP.
4. The PSP decrypts the key and puts it into the encrypted RAM of the VM.
5. Here it is picked up by the embedded grub bootloader (which was also part of the measurement) and passed to the Kernel.
6. The Kernel decrypts the combined boot/root OS image and mounts it.
7. The VM is now running in encrypted RAM space with encrypted OS images (boot, root, and home partitions) and the hardware and VMM operator never has access to the protected data in RAM or disk image.

The fully encrypted disk image needed some preparation to support the sequence of steps outlined above. It can be created as follows:

1. Extracting the root partition: Extract the complete root partition of an existing VM's image to a tar file. It needs to contain essential tools needed, such as cryptodisk and grub-efi.
2. Creating a two-partition raw image file: A new disk image file is generated, consisting of two partitions: the boot (UEFI) and the root/home partition.
3. Loopback mounting the image file: The newly created image file is mounted as a loop device, with 2 partitions: the EFI partition (p1) dedicated to the EFI system partition, and the Encrypted root partition (p2) used for the encrypted root filesystem.
4. Setting up EFI and cryptsetup: The EFI bootloader is configured to recognize the system's boot process and interact with the encrypted root partition. The root partition is encrypted using cryptsetup. During the initial setup, a simple password is used for the encryption (low-entropy or recovery password with many PBKDF2 iterations), however it can later be replaced by a high-entropy key for fast-booting, since grub is very slow in computing the PBKDF2 iterations.
5. Setting up grub for encrypted boot: grub configuration needs to be edited to enable boot from an encrypted partition.
6. Setting up fast-booting: to avoid 2 password prompts when booting the system (for boot and root partitions), a high-entropy password is added to cryptsetup. The initramfs configuration file is then edited, enabling the system to bypass the prompt for the encrypted root partition password.
7. Starting confidential VM: the initial ramdisk is rebuilt with all its partitions, and the system can boot using either the recovery or high-entropy password, enabling confidential computing with a fully encrypted disk image.

The process described in this section is based on previous work of the open-source community, [6], [7]. We found that many of the proposed patches are already present in an up-to-date Centos 9 Stream operating system. We had to modify and apply patches to some components, however, to make the whole system work (e.g. re-building OVMF with embedded grub, among other tasks).

5.3.4 Application Example: Secure Certificate Provisioning

To further deepen the reader's understanding of the trusted VM image creation and remote management processes outlined in the previous sections we would like to introduce a concrete application scenario for Confidential Computing in the project's context.

As has been outlined in our previous deliverables, relying parties (e.g. guest owner systems) should only send sensitive data to a remote system after successful remote attestation of the integrity of the remote system.

In this section we describe how our approach can be used to support trustworthy SSH (secure shell) and TLS (transport layer security) sessions involving trusted VMs running in the edge or public cloud.

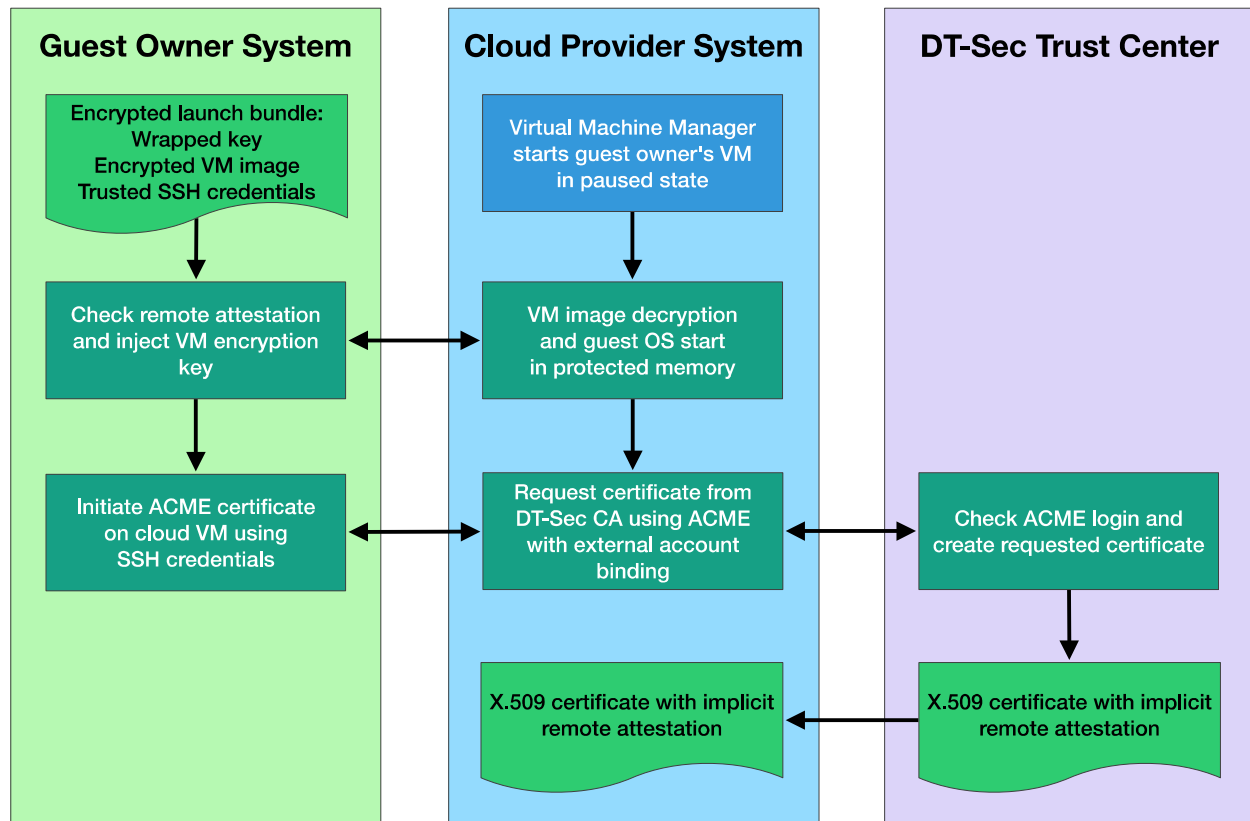


FIGURE 29: APPLICATION EXAMPLE - SECURE CERTIFICATE PROVISIONING

Figure 29 shows the three involved parties with their respective activities:

- On the guest owner system the initial image is prepared: It is encrypted with a disk key only known to the guest owner. Additionally, the guest prepares the SSH host key and loads software and data specific to the application.
- The cloud (or edge) provider receives the encrypted keys as well as parameters generated by the guest owner (the so-called “launch bundle”) to setup the VM in the VMM.
- The DT-Sec trust center is contacted by the trusted VM on first run (and later for certificate renewal) in order to create certificates.

The certificates originating from this process depend implicitly on the successful remote attestation between guest owner and cloud provider, because the guest must follow these steps:

1. Create an SSH host key in a trustworthy environment while setting up the image.
2. Start the VM only after successful remote attestation.
3. Log into the VM depending on the trusted host key. This way, the guest owner can be sure to log into a trusted environment.
4. Only then start the ACME client to request the certificate using the ACME login credentials (for external account binding). This step in turn depends on the trust into the SSH host key.

As a result, the trustworthiness of the certificate depends inherently on the successful remote attestation. Clients connecting to the application on this VM using TLS don't have to do the remote attestation step themselves, since this has been done already in the process of obtaining the certificate. Additionally, this approach enables standard SSH-based management of the remote VM without doing any additional attestation step.

5.4 Specification of Components

In this section we describe the tools and processes for guest owners to create individually encrypted, trusted VMs as well as the remote management and remote attestation/measurement facilities that we have implemented.

Using the Guest VM setup process, Guest owners can configure an encrypted VM template with their own keys. They can then provide the resulting encrypted image to the Cloud provider and initiate the startup of the VM in paused mode (necessary state for secure launch). Afterwards, Guest owners can initiate measured launch by checking the remote attestation from the Cloud provider and inject their VM's encryption key. The VM can then start in protected mode.

5.4.1 Setup Phase

5.4.1.1 Overview

The Guest VM Setup tool is a tool VM based on CentOS 9 allowing guests to create and set up their own encrypted VM.

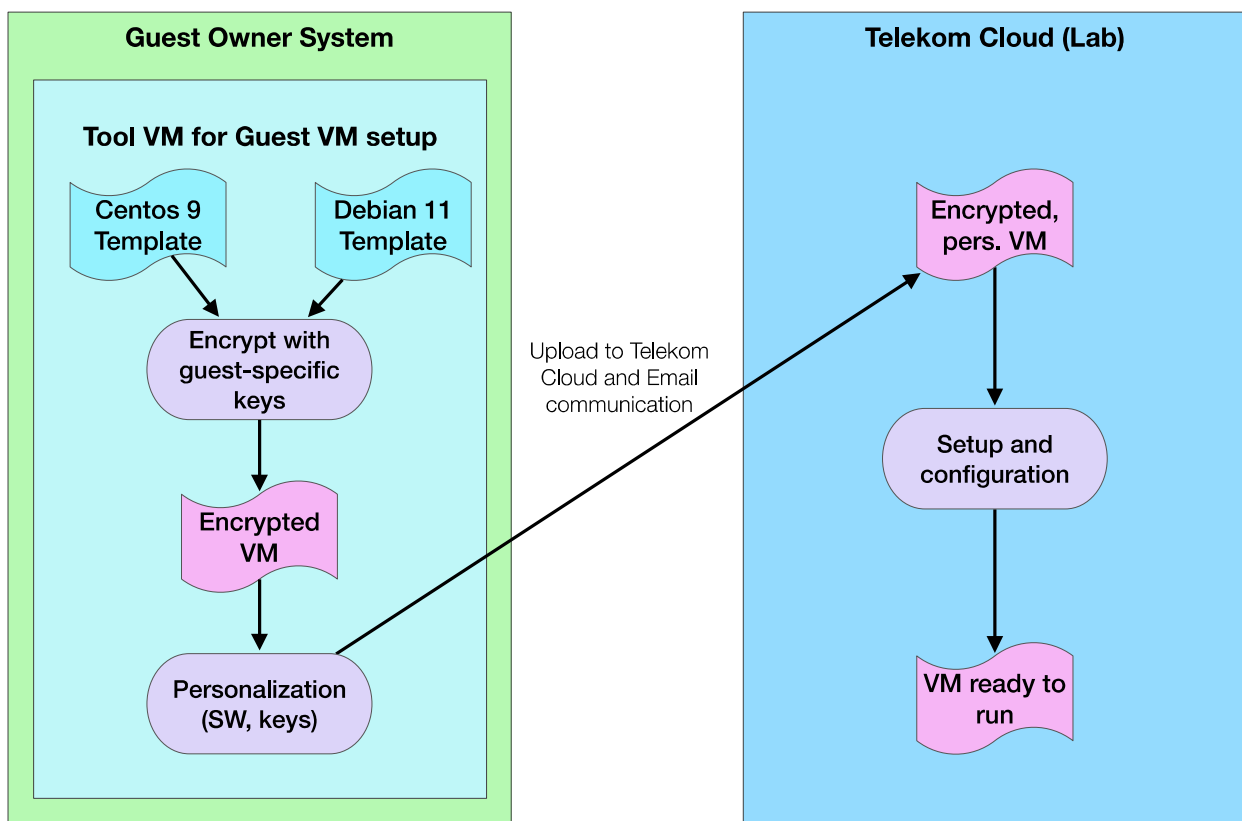


FIGURE 30: SETUP OF THE GUEST VM

Guests can set up their encrypted VM using the VM-tool provided by DT. There are currently 2 available OS templates for encrypted VM, Debian 11 and CentOS 9. After choosing a template, guests can encrypt it with their own keys to replace the DT default one. Afterwards, the VM image is transmitted to the Telekom Cloud Lab, to ensure that the encrypted VM is ready to run.

5.4.1.2 Commands Available in the VM for Guest VM Setup

The Guest VM Setup Tool contains all the necessary tools for guests to:

- select an encrypted VM image template (Debian 11 or CentOS 9),
- encrypt that image with 2 different keys (low entropy password as a recovery key, and high entropy password dynamically generated for faster boot of the VM), and
- mount the image on the tool VM (to perform administrative tasks, such as editing configuration, install updates and proprietary software, create an initial SSH key...).

The Guest VM Setup tool was originally supposed to be a docker container, but it was later replaced by a CentOS 9 based-VM, as it would have required too many privileges to run on Guest. Pre-configured VMs are available as well as the tools as a separate download for partners that want to use their own Guest setup VM.

The pre-configured images as well as intermediate files are located in directories below the /workdir directory in the VM root filesystem. The structure of the /workdir in the VM is as follows:

/workdir

images/

debian-template.qcow2 (with pre-defined recovery key)

centos9-template.qcow2 (with pre-defined recovery key)

transfer/

debian_XXXXX.qcow2 (encrypted with new keys)

centos_XXXXX.qcow2 (encrypted with new keys)

keys/

YYYYY_XXXXX_recovery-password.txt (user-given recovery password)

YYYYY_XXXXX_high-entropy-password.txt

Present in the original directory

Created by commands

All the commands described below are already set up in the Guest VM Setup tool. They need to be run with root privilege.

```
prepare [--debian/--centos] <VM name> <recovery-password>
```

This command is used to create a new encrypted image of a chosen template with personalized password. It works as follow:

- Arguments:
 - `--debian/--centos` : the chosen base template (CentOS 9 or Debian 11)
 - `<VM_name>`: the name of the created image. It will be renamed as “`template_name.qcow2`” (eg: “`centos_name.qcow2`”)
 - `<recovery-password>`: password of choice which will be used to encrypt the image
- Generate a high entropy password, store recovery password and high-entropy password in the “`/workdir/keys` directory”.
- Make a copy of the chosen template to “`/workdir/transfer`” directory and rename it
- Encrypt the image with both recovery-password and high-entropy password using `cryptsetup`.

```
mount <VM name>
```

This command is used to mount the encrypted VM image from the `/workdir/transfer` directory and give the user a command line in the VM `/root` directory. It works as follow:

- Argument:
 - `<VM_name>`: name of the image to mount, can be written either as “`template_name`” or “`name`” (eg : “`centos_name`” or “`name`”).
- Connect the image as NBD (Network Block Device) to a free NBD device (located in “`/dev/nbd*`”).
- Open the encrypted luks partition using the recovery password in “`/workdir/keys`” directory.
- Mount the associated cryptsetup device (“`dev/mapper/encrypted-image`”) and all its contained directories at “`/mnt/root`”.
- Give the user a command line from “`/mnt/root`” using `chroot`.

```
unmount-images
```

This command is used to unmount and disconnect every image.

- Unmount all partitions.
- Close every cryptsetup devices.
- Disconnect every NBD device.

```
reset-images
```

This command is used to reset the VM setup tool environment. Use this command with care since it deletes all image personalizations.

- Unmount all images (the same as `umount-images` command).
- Delete all previously created files in the `workdir` (`/workdir/transfer` containing created images, `/workdir/keys` containing keyfiles).

The complete REST API is described in appendix A.9.

5.4.2 Operation Phase

After the prepared image from the setup phase has been transferred to the host system in the DT-Sec Berlin lab and pre-configured by the support personnel, the guest owner needs to perform a few last steps to prepare the confidential remote operation, after which he or she can start using the remote computing resource.

5.4.2.1 Overview

Before the actual operation phase, the following two steps must be performed by the guest owner:

- Creating a so-called “launch bundle”, consisting of AMD-SEV-related key material for remote management.
- Sending two public keys from this launch bundle to the remote system via the REST API described below.
- Notifying the host admin about the completion of these two steps.

After these steps the remote VM is ready to run.

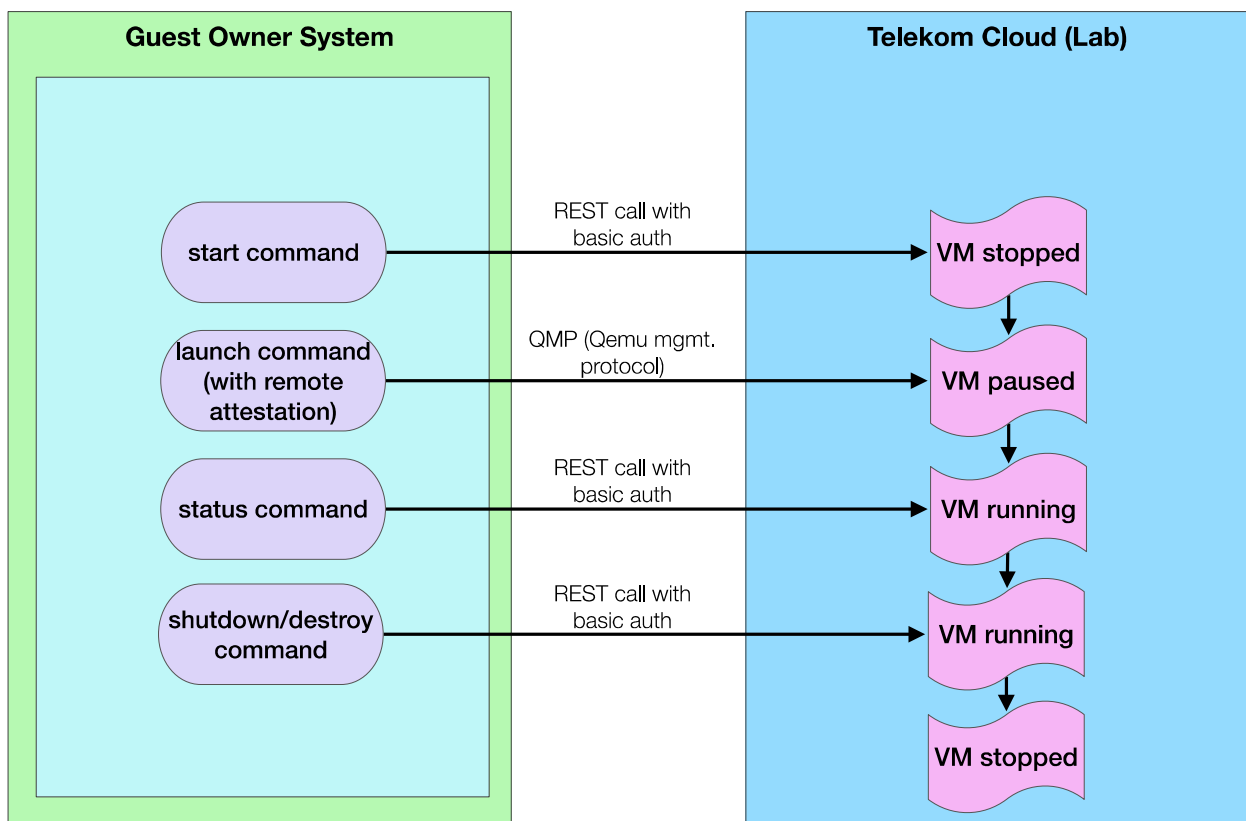


FIGURE 31: MANAGING THE GUEST VM

Figure 31 shows the available remote management commands as well as the life cycle of the VM:

- With the start command the guest owner starts the remote VM in paused mode.
- The launch command performs the actual remote attestation. If the remote attestation passes (i.e. the measurement is equal to the locally computed value) the launch command also transmits the encrypted disk key to the remote VM. The VM then starts and is ready to perform its function.
- At any time, the guest owner can request the status of the remote VM using the status command.
- When the VM is not needed anymore, the guest owner should issue a shutdown command, followed later by a destroy command to stop and switch off the remote VM. Note: destroy should be issued about 30 s after the stop command to allow for clean shutdown. Some VMs need the destroy command because the Linux kernel is not switching off the VM properly. Best practice would be to check with the “status” command and issue the destroy only if the VM is still running after 30 s.

Detailed configuration steps are given in appendix A and in README files on our download site.

6 QUALITY OF EXPERIENCE EVALUATION

In recent years, immersive video delivery has improved significantly. This type of video content can be viewed in XR, including VR, AR, and MR, to provide near-lifelike 3D objects and scenes. To represent these 3D entities, PCs are commonly used as the format providing high-fidelity representations without any constraint on the viewpoint and interaction. A PC comprises thousands or even millions of points, including information about colours (e.g., RGB) and geometry (x,y,z coordinates) of each point. Thus, the usage of PCs costs a large amount of storage and network bandwidth. Efficient ways to compress PCs are of importance to solve this issue. The compressed PC is then stored and/or delivered to end users via video streaming techniques (e.g., WebRTC, and HAS [8]).

In HAS, the content is encoded into various quality levels, then temporally split into multiple segments with the same duration. These segments are stored on one or multiple servers in a content delivery network [9]. The end users' media player selects the quality level for each segment based on the network conditions (i.e., observed throughput), the devices' characteristics (i.e., resolution and buffer), and the users' preferences. On the other hand, WebRTC makes its own decisions about the target encoding bitrate based on the estimated throughput, at which the video encoder is called for video compression. In both cases, the video quality can be changed in a streaming session, which leads to quality switches. The quality switches are well-known to affect the QoE in traditional video streaming [10]. However, they have not been fully considered in dynamic point cloud streaming.

Due to extremely high data volumes of uncompressed PCs, PCC is necessary to reduce both storage requirements and the amount of data delivered through networks. Initial studies started with compression of static 3D objects [11] [12] [13]. However, recent work focused on dynamic scenarios [14]. The MPEG developed standardized solutions for point cloud compression by leveraging their existing codecs, such as HEVC [15]. Their video point cloud encoder implementation can achieve a compression rate of 125:1. However, PCC comes at the cost of visual quality, determined by the QP. A higher QP provides a lower bitrate but leads to a lower quality.

Therefore, understanding the impact of different factors in video streaming techniques and PCC on the QoE is of importance. PCs have been evaluated in different viewing conditions (i.e., VR HMD and 2D screens) [16] [17]. However, research on the subjective quality assessment of PCs in AR environments is still limited. AR enhances people's perception of physical and virtual environments [18]. Thus, it is an interesting setting for immersive telepresence applications, which we develop and assess in the SPIRIT project.

Most of the effort in building QoE tools has been focused on 2D videos. Petrangeli et al. [19] proposed a parametric statistical model computing QoE based on average bitrate, its standard deviation, stall duration, and stall frequency. Tran et al. [20] introduced BiQPS using a Long-Short Term Memory network to predict the QoE. BiQPS takes into account stall duration, QP, bitrate, resolution, and frame rate. Recently, the ITU defined a standardized QoE model, the P.1203 model [21] for 2D video streaming that is trained and validated with different large databases of QoE. Some effort has been made to predict QoE in point cloud streaming on the basis of mathematical models, such as [22]. However, these models are often validated by subjective tests in VR environments.

UNI-KLU, in collaboration with imec, conducted a subjective test to assess the impact of quality, quality switching, viewing distance, and content characteristics on the perception of point clouds in AR environments. The output of this work includes (i) a platform for subjective quality assessment in AR environments, (ii) a dataset of rating scores that can be used for training and validating future QoE models as well as the results (findings) of the subjective tests that produced these rating scores, and (iii) a machine learning based QoE model. In addition, the

work also fine-tuned an existing ITU QoE model – that has been originally developed for conventional 2D videos – to predict the QoE for point clouds in AR environments. The details are provided in the following subsections.

The software, dataset, results (findings) from the subjective test, and the QoE models can be used by the project partners and by Open Call participants in use cases (applications) involving point cloud content in AR environments for three purposes:

- to build their own preview and subjective test software informing their use cases;
- to be guided by the subjective test results when deciding about the quality level(s) of the PC content to be captured, transmitted, and presented in their use cases; and
- to predict how their users will perceive the PC content in AR environments.

6.1 A PLATFORM FOR SUBJECTIVE QUALITY ASSESSMENT OF POINT CLOUDS IN AUGMENTED-REALITY ENVIRONMENTS

We introduced a new platform for subjective quality assessment of PCs in AR environments. The platform provides multiple options for configuring the rendering of dynamic PC objects and meshes, including changing the content, quality, viewing distance, and representation, as well as previewing and interacting with the PC objects. Additionally, the proposed platform can be deployed to create subjective tests of the visual quality of dynamic PCs and meshes in AR environments. Our platform is first presented in [23].

The platform, which is published on GitHub at

➔ <https://github.com/shivivats-aau/MR-Subjective-Testing-Platform>,

is built using Unity². The MRTK 2³ from Microsoft is utilised to work with the HoloLens 2. Despite the HoloLens 2 having standalone capabilities, we use Holographic Remoting⁴ from MRTK 2 to run the platform on a workstation while taking input from and sending output to the HoloLens 2. The architecture is shown in Figure 32. The platform has two major functionalities: (i) point cloud previewing and (ii) subjective testing.

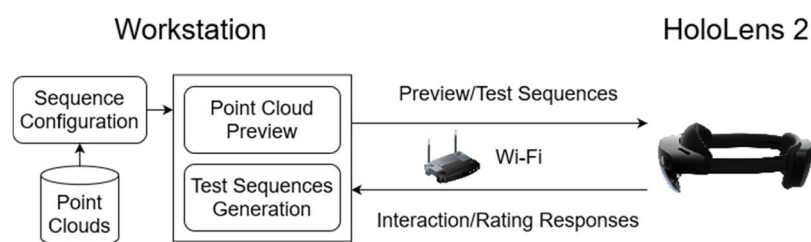


FIGURE 32: PLATFORM ARCHITECTURE FOR SUBJECTIVE QUALITY ASSESSMENT

6.1.1 Point Cloud Previews

The purpose of the preview functionality is to allow the tool users to explore and compare various configurations of the point clouds. These configurations are controlled via a menu in the HoloLens 2 device. Multiple objects can be added and viewed side by side, and each object

² Version 2021.3.19f1. <https://unity.com/>. Accessed: 20 April 2023.

³ <https://github.com/microsoft/MixedRealityToolkit-Unity>. Accessed: 20 April 2023.

⁴ <https://learn.microsoft.com/en-us/windows/mixed-reality/develop/native/holographic-remoting-overview>. Accessed: 20 April 2023.

can be configured individually. As shown in Figure 33, an object can be configured in terms of the following properties: *Representation*, *Viewing Distance*, and *Quality*. The *Loot* object (front) is rendered using the *Point* representation at a distance of 3 m from the viewer with quality level 3. The *Soldier* object (at the back) is rendered using the *Square* representation at 5 m distance with the same quality.

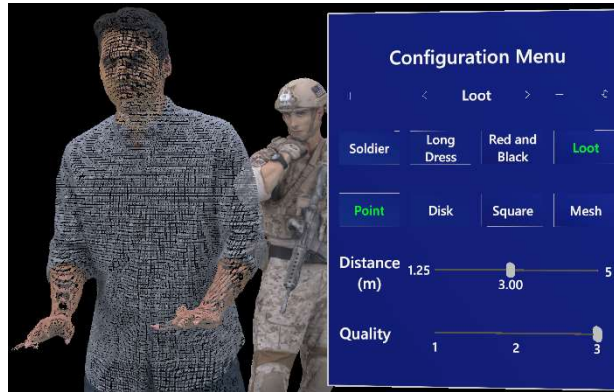


FIGURE 33: CONFIGURATION CONTROL PANEL AND PC OBJECTS

6.1.2 Subjective Testing

The second major functionality of our platform is the ability to compose and perform subjective tests within the given parameter ranges. These are single-stimulus tests designed with the purpose of testing the impact of various factors, such as distance and quality, on the perceived quality of the object and, thus, the QoE of the test participant. These tests can be configured using the Unity UI and are divided into “tasks”. Within a task, the tool user (researcher or test director) can select several configurations for the point clouds, similar to the configuration options in the preview. All the possible permutations from the chosen configurations are determined and displayed to the test participant in random order. Randomization removes any bias the participants might obtain by watching the sequences in a particular order. The Unity UI for the subjective test configuration is shown in Figure 34.

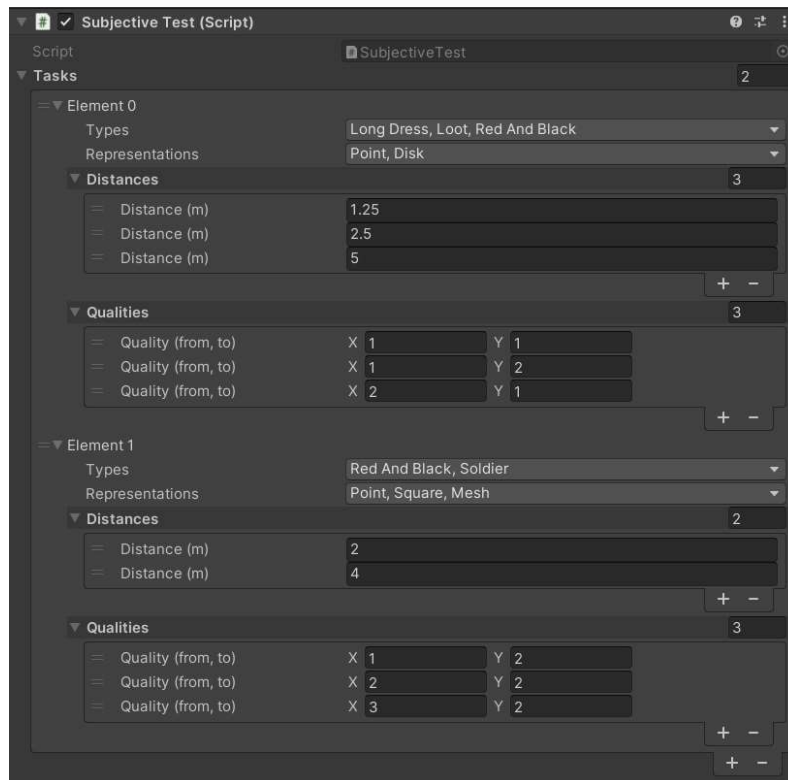


FIGURE 34: SUBJECTIVE TEST CONFIGURATION UI

The test workflow is designed so that the participant is asked to rate the perceptual quality of each sequence after watching it. This is made possible via an immersive slider which allows the participants to rate the quality from 1 through 10 with textual guidelines to support the ratings (i.e., 1, 2 – very bad, 3, 4 – bad, 5, 6 – fair, 7, 8 – good, 9, 10 – very good). These ratings are stored in a CSV file with a unique numerical ID assigned to the test participant and a string describing the sequence they just watched. The timestamp of this action is also stored and can be used to synchronize with a separate questionnaire that the participants might fill in, among other usages.

Furthermore, while the test participants perform the test using the HMD, the tests can be monitored by simply looking at the Unity application on a computer screen. This includes tracking the participant's progress and any difficulties they might have during the test.

6.2 DATASET AND RESULTS OF SUBJECTIVE TESTS FOR POINT CLOUDS IN AUGMENTED-REALITY ENVIRONMENTS

6.2.1 Experiment Tasks

As this work focuses on the usage of PCs in telepresence applications, we used four PC video sequences from the *8i Voxelized Full Bodies Database* [24]: *Loot*, *RedAndBlack*, *LongDress*, and *Soldier* as shown in Figure 35. The first two have a lower contrast content than the other two [16]. Each sequence captures a complete object using 42 RGB cameras operating at 30 fps for 10 seconds. We use the MPEG V-PCC reference software TMC2⁵ to create compressed PCs by varying the QPs. This software already includes five sets of QPs defined in

⁵ <https://github.com/MPEGGroup/mpeg-pcc-tmc2>. Accessed 16 November 2023.

MPEG’s CTC [25] with the G-QP and T-QP ranging from 16 to 32 and from 22 to 42, respectively. Three such pairs from the MPEG PCC software with the lowest, middle, and highest QPs (i.e., the rates R5, R3, and R1 in MPEG’s software TMC2) are selected as follows:

- Q1 (R1): (G-QP, T-QP) = (32, 42)
- Q2 (R3): (G-QP, T-QP) = (24, 32)
- Q3 (R5): (G-QP, T-QP) = (16, 22)

Q3 is thus the best quality level. The bitrate of the objects decreases with increasing QPs (see Table 7).



FIGURE 35: TEST OBJECTS IN 8I VOXELIZED FULL BODIES DATABASE

TABLE 7: BITRATES IN MBIT/S OF DIFFERENT QUALITY LEVELS OF THE PC OBJECTS

Video	Quality		
	Q1	Q2	Q3
Loot	2.28	5.63	16.68
LongDress	4.64	14.05	46.78
RedAndBlack	3.39	7.55	22.90
Soldier	4.38	11.58	35.29

We designed two tasks for each participant, with both tasks consisting of 18 sequences of length 10 s. Table 8 describes the sequences. Before the experiment, participants are asked to provide some background information, including age, gender, eyesight, and experience in viewing VR, AR, and MR content.

TABLE 8: NOTATION AND DESCRIPTION OF THE TEST SEQUENCES

Task	Notation	Description	Sequences
1	Q_{ij}	The video starts with quality Q_i , then switches to Q_j after 5 s.	<i>Loot</i> and <i>LongDress</i>
2	$Q_i_D_k$	The video is watched at quality Q_i at distance D_k .	<i>RedAndBlack</i> and <i>Soldier</i>

6.2.1.1 Task 1: Impact of Video Encoding and Quality Switches

The participant watches nine sequences for each of the two objects, including three sequences with static quality (Q_1 , Q_2 , and Q_3) and six sequences with a quality switch in the middle of each sequence. The objects are placed 5 m from the participant so that the whole body can be viewed. *Loot* and *LongDress* are used in this task as they belong to low and high contrast levels, respectively.

6.2.1.2 Task 2: Impact of Viewing Distance

The participant watches static-quality sequences of the other two objects (*RedAndBlack* and *Soldier*) at quality levels Q_1 , Q_2 , and Q_3 at three distances:

- D_1 : 1.25 m (only face and shoulder in FoV)
- D_2 : 2.5 m (only upper body in FoV)
- D_3 : 5 m (full body in FoV)

The order of tasks and sequences for each task are randomized for each participant. After watching each sequence, the participant is asked to rate the perceptual quality (i.e., 1, 2 – very bad, 3, 4 – bad, 5, 6 – fair, 7, 8 – good, 9, 10 – very good) through the immersive slider shown in Figure 36.

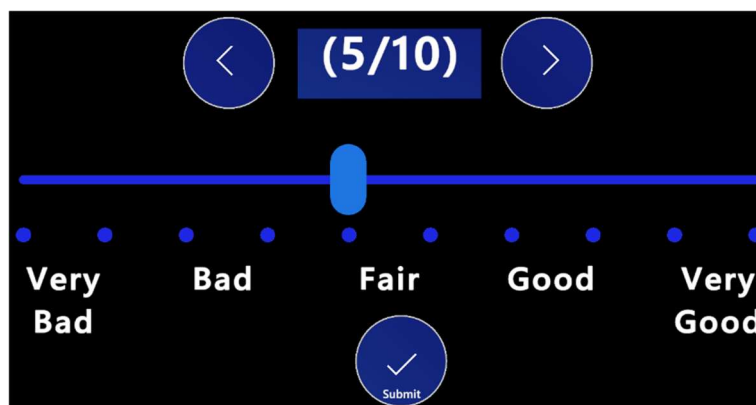


FIGURE 36: IMMERSIVE RATING SLIDER WITHIN THE USER INTERFACE OF THE HOLOLENS 2 AS USED DURING THE SUBJECTIVE TESTS

After the experiment, participants are asked to provide feedback on their experience regarding levels of general discomfort, nausea, sweating, headache, or dizziness that they may have experienced. Participants also answer a question of whether they feel the PC objects are part

of the real environment by selecting one of five options: (i) strongly disagree, (ii) disagree, (iv) neutral, (j) agree, and (v) strongly agree. The total duration of a single experiment is approximately 25 minutes.

6.2.2 Dataset Description

Our dataset of the subjective tests is published on GitHub:

➔ <https://github.com/minhkstn/QoE-and-Immersion-of-Dynamic-Point-Cloud>.

The dataset includes two files: `rating_score.csv` and `questionnaire_responses.csv`. This section describes the content of each file.

6.2.2.1 Rating Scores

Table 9 describes the structure of the data in file `rating_scores.csv`.

TABLE 9: STRUCTURE OF RATING SCORES DATABASE

Column Name	Data Type	Description
<code>participant</code>	Int	The index of the participant
<code>object</code>	String	The name of the 3D object in the test sequence
<code>start_quality</code>	String	The quality in the first 5 s of the test sequence
<code>end_quality</code>	String	The quality in the last 5 s of the test sequence
<code>distance</code>	Float	The viewing distance (m)
<code>start_gqp</code>	Int	The G-QP in the first 5 s of the test sequence
<code>start_tqp</code>	Int	The T-QP in the first 5 s of the test sequence
<code>end_gqp</code>	Int	The G-QP in the last 5 s of the test sequence
<code>end_tqp</code>	Int	The T-QP in the last 5 s of the test sequence
<code>qoe</code>	Int	The score rated by the participant

6.2.2.2 Questionnaire Responses

Before the test, we asked the participants to provide some background information, and tested their colour vision with an Ishihara test⁶.

⁶ https://en.wikipedia.org/wiki/Ishihara_test. Accessed 03 November 20223.

After the test, the participants answered a survey about cybersickness and the immersion level of the tested objects. The answers are stored in the table of file `questionnaire_responses.csv`. The structure of the table is described in Table 10.

TABLE 10: STRUCTURE OF QUESTIONNAIRE ANSWERS

Column Name	Data Type	Description
Participant	Int	The index of the participant
Which number do you see below?	Int	The first tested number for Ishihara test (must be 12)
Which number do you see below?	Int	The second tested number for Ishihara test (must be 45)
Which number do you see below?	Int	The third tested number for Ishihara test (must be 74)
During the assignment I felt... [General discomfort]	String	Level of agreement with options: Strongly disagree, Disagree, Neutral, Agree, Strongly agree
During the assignment I felt... [Nausea]	String	Level of agreement with options: Strongly disagree, Disagree, Neutral, Agree, Strongly agree
During the assignment I felt... [Sweating]	String	Level of agreement with options: Strongly disagree, Disagree, Neutral, Agree, Strongly agree
During the assignment I felt... [Headache]	String	Level of agreement with options: Strongly disagree, Disagree, Neutral, Agree, Strongly agree
During the assignment I felt... [Dizziness]	String	Level of agreement with options: Strongly disagree, Disagree, Neutral, Agree, Strongly agree
How would you describe the general experience? [I felt the objects were part of the real environment]	String	Level of agreement with options: Strongly disagree, Disagree, Neutral, Agree, Strongly agree
What is your gender?	String	Gender with options: Male, Female, Non-binary, Prefer not to say
What is your age group?	String	Age group with options: Under 18, 18-24, 25-34, 35-44, 45-54, 55-64, 65-74, 75-84, 85 or older

Column Name	Data Type	Description
How often have you watched/experienced extended reality content (e.g, 360 videos/games, holographic content)?	String	Frequency with options: Never, Once, A few times, Monthly, Weekly, Daily

6.2.3 Subjective Test Results

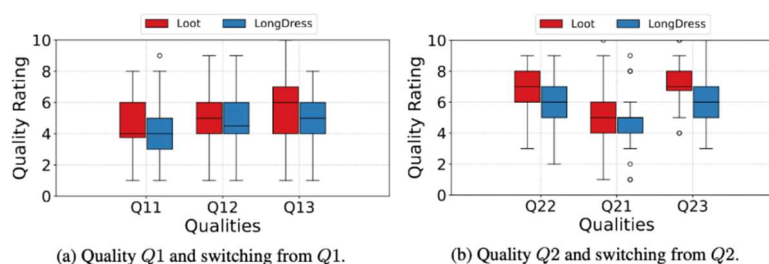
In this section, we provide an overview of the participants and present and analyse the subjective test results. We published our analysis in [26] [27].

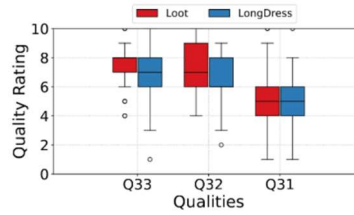
6.2.3.1 Participants

A total of 36 participants, who were recruited from the UNI-KLU, attended the subjective test, including 22 (61%) males, 13 (36%) females, and 1 (3%) non-binary. 3 (8%) were in the age group of 18 to 24 years, 18 (50%) were between 25 and 34, 12 (33%) between 34 and 44, 2 (6%) between 45 and 54, and 1 (3%) between 55 and 64. The colour vision of the participants is evaluated using the Ishihara test [28]. Four participants failed this test, so their ratings are excluded. Hence, the results in this section are gathered from 32 participants which is compliant with ITU-T P.919 [29].

6.2.3.2 Impact of Video Encoding

Figure 37 shows the quality ratings of the participants for the test sequences at different quality levels and quality switches. Regarding the sequences with static quality levels (i.e., Q11, Q22, Q33 in Figure 37(a)-(c)), it can be seen that objects encoded with lower qualities have lower scores. At least 75% of the viewers gave *Loot* (*LongDress*) a rating of 6 (5) or less for the lowest quality sequence, Q11. Their medians are both 4, which means a bad experience. With a higher-quality sequence, Q22, the median quality scores improve remarkably to 7 (i.e., good) and 6 (i.e., fair) for *Loot* and *LongDress*, respectively. For the highest-quality sequence, Q33, there is an improvement in quality ratings, but it is less remarkable than for Q22. *Loot* still receives good ratings from participants, with a median of 8 (i.e., good), while *LongDress* achieves ratings ranging from fair (median of 6) for Q22 to good (median of 7) for Q33. Furthermore, though Q33 can achieve very good ratings (9 or 10), the majority (at least 75%) of the participants rate this sequence at no more than 8. To statistically validate these claims, we used one-way ANOVA [30] and post-hoc comparison analysis using Tukey's HSD test [31]. According to the ANOVA results, there is a significant difference ($p < 0.001$) between the three quality levels. Post-hoc pairwise, Tukey's HSD reveals that quality ratings do not differ significantly ($p > 0.05$) between Q22 and Q33 for *Loot*, but do for *LongDress* ($p < 0.05$). Furthermore, there are significant p-values ($p < 0.001$) between Q11 and the others.





(c) Quality Q3 and switching from Q3.

FIGURE 37: QUALITY RATINGS FOR DIFFERENT QUALITY LEVELS AND QUALITY SWITCHES

6.2.3.3 Impact of Quality Switches

Figure 37 also describes the participant ratings for different quality switches, including switching up when the quality is increased and switching down when the quality is decreased. There is no remarkable improvement in the quality scores when the sequence starts at quality Q1 (i.e., Q11, Q12, and Q13). ANOVA analysis indicates no significant difference ($p > 0.05$) among the quality scores for both *Loot* ($p = 0.07$) and *LongDress* ($p = 0.13$). This can be attributed to the severe distortion of Q1 in the initial 5 s that affects the QoE when watching the entire 10 s video. Regarding switching down, when the quality changes from Q2 or Q3 to Q1 (i.e., Q21 or Q31), the quality ratings are markedly reduced compared to the constant-quality sequences (Q22 and Q33). However, there are no significant differences when the quality changes between Q2 and Q3. We conducted a paired samples t-test [32] to further validate this observation. It shows non-significant p-values between Q22 and Q23 (e.g., $p = 0.6136$ for *Loot*) as well as between Q33 and Q32 (e.g., $p = 0.1162$ for *LongDress*).

Combined with the results in the previous section, we claim that the end user hardly recognizes the quality differences between Q2 and Q3. Thus, we recommend that it is unnecessary to change the quality from Q2 to Q3 when the object is viewed at a distance of 5 m. This can remarkably reduce the amount of transferred data.

6.2.3.4 Impact of Viewing Distance

Figure 38 shows the quality ratings of the test sequences at different viewing distances. It is noticeable that distance significantly impacts the visual quality of the objects: the higher the viewing distance, the higher the quality scores. The reason is that, at a higher distance, it is harder for the viewers to recognize some quality distortions; thus, they give higher quality scores, which is comparable to what has been reported for traditional video sequences [33]. Additionally, we observe that to achieve the same visual quality, the object should be encoded with lower QPs (i.e., more data) if viewed closer. For example, *RedAndBlack* at quality Q1 is rated on average 4.8 at 5 m, and this object must be encoded at Q2 to gain a similar score (i.e., 4.9) if it is viewed at 1.25 m ($p = 0.5$ in a paired t-test).

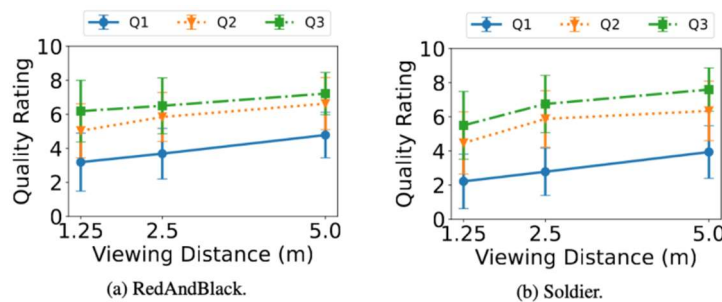


FIGURE 38: AVERAGE QUALITY RATINGS FOR DIFFERENT DISTANCES

6.2.3.5 Impact of Content Characteristics

We also evaluate the impact of content characteristics on the visual perception of participants for both tasks, as shown in Figure 39. *Loot* and *RedAndBlack* achieve higher quality ratings in most cases. For example, the quality scores of *Loot* and *RedAndBlack* with quality Q1 viewed at distance D3 (i.e., 5 m) are 4.5 and 4.8, respectively. Under the same conditions, these figures for *LongDress* and *Soldier* are 4.2 and 3.9, respectively. This can be explained by the fact that participants are less sensitive to quality distortion/changes for the content with fewer contrast differences. This finding extends the results presented in the work [16] on 2D screens to an AR environment with AR HMDs, in which the texture of the objects is a crucial factor for viewers.



FIGURE 39: AVERAGE QUALITY RATINGS OF PARTICIPANTS. IT SHOULD BE NOTED THAT THE SEQUENCE QII IN TASK 1 IS EQUIVALENT TO QI_D3 ($i \in \{1, 2, 3\}$) IN TASK 2, ENCODED AT QUALITY QI AND VIEWED AT 5 M.

6.2.3.6 Cybersickness in AR

The cybersickness levels of the participants are illustrated in Figure 40. Figure 40a shows that most of the participants did not feel symptoms of cybersickness in their experiment session that lasted about 25 minutes. 84% and 81% of them did not sweat or feel nauseated, respectively. The most common symptom is dizziness, but only 21% of the participants reported feeling dizzy during the test. Figure 40b provides more details about the symptoms of the participants who received cybersickness. No one suffers from all the symptoms mentioned above. There is only one person who experiences three symptoms, including sweating, headache, and dizziness. Three participants felt two symptoms, and six others received one symptom. On the contrary, in a similar-duration subjective test [34] where participants were watching videos with four characters in a room and dolphins in the ocean with VR HMDs, cybersickness was a serious problem that affected more than 90% of the viewers.

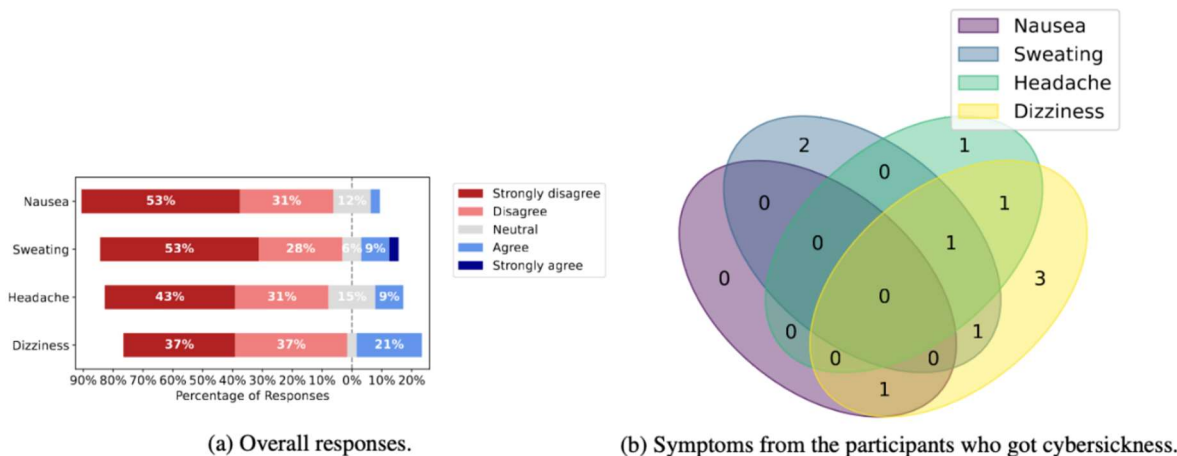


FIGURE 40: CYBERSICKNESS LEVELS OF THE PARTICIPANTS

6.2.3.7 Objects' Immersion Levels

Figure 41 shows the immersion levels of digital objects in the physical world rated by the participants. It can be seen from the figure that 39% of the participants (strongly) agreed that the objects were part of the real environment. Only 27% of the participants (strongly) disagreed with this feeling. Therefore, the tested objects and HoloLens 2 provide the feeling of telepresence to some extent. However, some participants complained about the quality of some parts of the objects, even at the highest quality level. For example, the hair of *RedAndBlack* was perceived as blocky, and the heels of *LongDress* were missing in some frames (see Figure 35). When we consider the impact of the participants' frequency of watching XR contents on the immersion level rating of the tested objects, there are two findings to be noted. First, most participants (5 out of 7 people) who have never watched XR content do not feel that the test sequences are real. Second, most experienced participants felt neutral or agreed that the objects were real. These people may have a good understanding of how 3D objects look in such environments, and thus may have lower expectations in terms of feeling the presence of these digital objects.

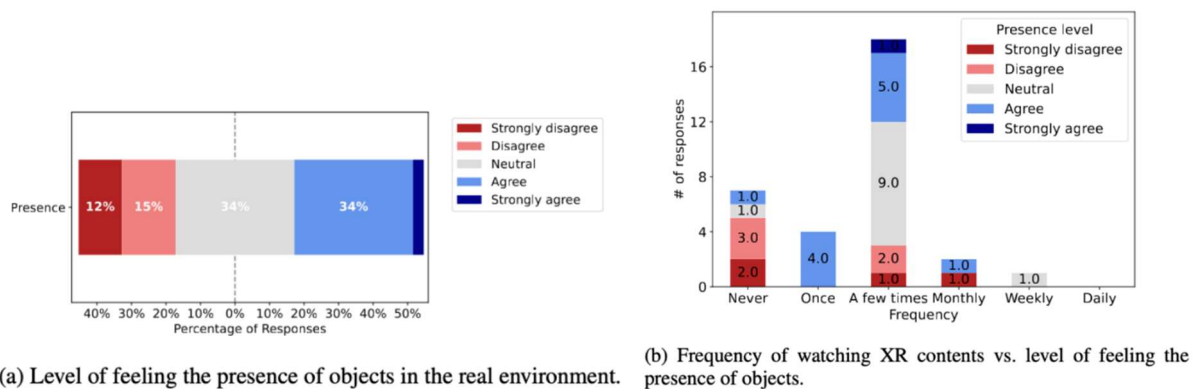


FIGURE 41: OBJECTS' IMMERSION LEVELS

6.3 QOE MODELS

Using the dataset in Section 6.2, we evaluate the performance of supervised machine learning techniques in predicting the QoE for point clouds in AR environments. In addition, we also fine-tuned an existing QoE model, which was originally designed for 2D videos, called ITU-T P.1203 mode 0, to predict the QoE of point clouds.

These models can be integrated in the testbed of Open Call participants to predict the QoE of the end users.

6.3.1 Machine Learning-Based QoE Models

6.3.1.1 Data Preparation

We consider four influence factors, including encoding parameters, quality switching, viewing distance, and content characteristics. The first two factors are represented by the values of start and end QPs of sequences. The content characteristics can be represented by the bitrate of the encoded bitstream. As observed in our test, the objects have different bitrates even at the same quality level (same QPs). For this evaluation, each input data record comprises six features: *start G-QP*, *start T-QP*, *end G-QP*, *end T-QP*, *viewing distance*, and *bitrate*. The corresponding ratings of the participants are used as the learning targets. We received 1152 responses in total from 32 participants. After omitting outliers defined by the interquartile range

method, 1107 responses are used as the input data. To receive a reliable and unbiased estimate of model performance, we use leave-one-out cross-validation. The input data is split into k groups, in which $k - 1$ groups are used as the training dataset, and the remaining one as the testing dataset. The process of splitting the data is repeated k times so that every group is used as a testing dataset once. There are, in total, 36 test sequences; hence, we have $k = 36$ so that the ratings for 35 test sequences are used for training, and the others are for testing.

6.3.1.2 Evaluation Results

Here, we train and evaluate common machine learning model for QoE prediction in AR environments. Their performance is reported in Table 11.

TABLE 11: PERFORMANCE OF MACHINE LEARNING MODELS IN PREDICTING THE QOE OF POINT CLOUDS IN AN ENVIRONMENTS. THE BOLD ENTRY SIGNIFIES THE BEST PERFORMANCE.

QoE Model Type	R2 Score	MSE
Gradient Boosting Regressor	0.8582	0.2874
Random Forest Regressor	0.8384	0.3273
Decision Tree Classifier	0.8155	0.3738
Decision Tree Regressor	0.7954	0.4146
Polynomial Regression (Degree 2)	0.7650	0.4761

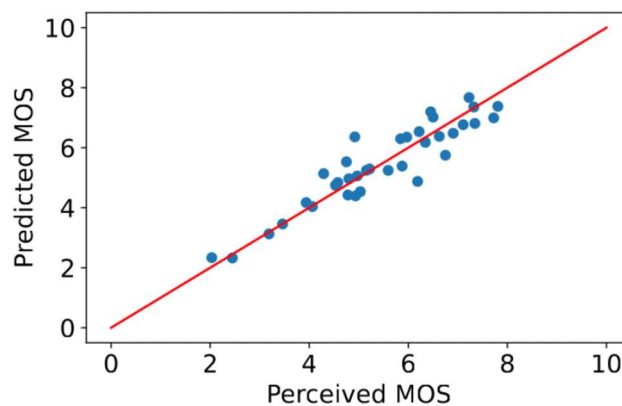


FIGURE 42: PERCEIVED MOS (FROM OUR SUBJECTIVE TEST) VERSUS PREDICTED MOS USING THE GRADIENT BOOSTING REGRESSOR. THE RED LINE REPRESENTS THE $Y = X$ LINE.

Figure 42 shows the correlation of the perceived MOS (from our subjective test) with respect to the predicted MOS using the Gradient Boosting Regressor. The predicted MOS is highly correlated (Pearson correlation coefficient = 0.93) with the perceived MOS, which was gathered from our subjective test.

Figure 43 presents feature importance scores of input features in the Gradient Boosting Regressor model using the Python scikit-learn library. A higher score means more importance



when building a predictive model. It is highlighted that the end T-QP plays the most crucial role for the Gradient Boosting Regressor in predicting the QoE, followed by viewing distance and end G-QP. Their importance scores are 0.32, 0.24, and 0.19, respectively. The content characteristics represented by the bitrate are the least relevant feature of the prediction model, with an importance score of 0.05.

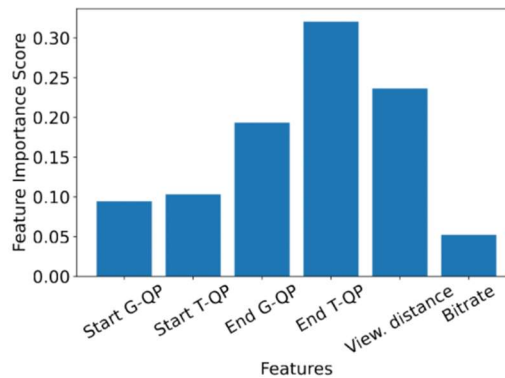


FIGURE 43: FEATURE IMPORTANCE SCORES OF INPUT FEATURES IN GRADIENT BOOSTING REGRESSOR.

6.3.2 Fine-tuned P.1203 Mode 0 Model

The ITU put in a great deal of effort in video quality estimation models, namely ITU-T P.1203.¹⁷ This P.1203 model was implemented and published on GitHub⁸. The model's inputs include video characteristics (i.e., bitrate, framerate, codec, and frame size), streaming parameters (i.e., stall events), and viewing conditions (i.e., device type and viewing distance). Point cloud streaming also shares some parameters that can be used in the P.1203 model, such as bitrate, framerate, stall events, and viewing distance. However, as the coefficients in the original P.1203 model were determined from a training phase based on a subjective database for 2D videos, they need to be re-trained with a new subjective database for point cloud streaming.

We split this dataset into a training set and a validation set. We train the coefficients of the P.1203 model with the former set and validate its performance with the latter one. The results show that our fine-tuned P.1203 model outperforms the original model from the ITU. Our model achieves an RMSE of 0.813, compared to 0.887 of the original P.1203 model with the training set. The PLCC and SRCC of our fine-tuned model are also significantly higher than that of ITU's model (see Table 12).

TABLE 12: PERFORMANCE OF THE ORIGINAL P.1203 MODE 0 AND OUR FINE-TUNED P.1203 WITH TRAINING AND VALIDATION DATASET.

	Training			Validation		
	PLCC ↑	SRCC ↑	RMSE ↓	PLCC ↑	SRCC ↑	RMSE ↓
ITU P.1203	0.766	0.785	0.887	0.918	0.829	1.032

⁷ <https://www.itu.int/rec/T-REC-P.1203.1/en>. Accessed: 02 November 2023.

⁸ <https://github.com/itu-p1203/itu-p1203>. Accessed: 16 November 2023.

	Training			Validation		
Fine-tuned ITU P.1203	0.919	0.953	0.813	0.958	0.828	0.955

The fine-tuned P.1203 model is published in

➡ <https://github.com/minhkstn/itu-p1203-point-clouds>.

This work is accepted by the Mile-High Video 2024 Conference [35].

7 CONCLUSIONS

As we anticipate forthcoming technical refinements and iterations, this conclusion serves as a checkpoint rather than a final destination. The document lays the foundation for future enhancements and updates, emphasizing the dynamic nature of the project.

The initial work consisted of integrating the partner components into a common testbed environment to enable third parties to use the various partner components and the testbeds for their use cases.

In the next step, the two testbeds will be connected so that the use cases, application frameworks can be deployed and validated across the testbeds. Several additional components, already identified, developed and presented in D3.1 [2] are prime candidates for integration in the platforms as well.

In parallel, an Open Call will be launched inviting third parties to realize their telepresence use cases experimentally. The components described in this document are available for this purpose and can be integrated into their use cases. The results from the experiments will be used by the project to improve the existing partner components and to develop new innovative components or connections.

As the project progresses, additional innovative components will be developed to enable and improve further telepresence applications. These innovative components are continuously integrated into the platform, tested and validated. Third-party components, in particular, may have a positive impact on the platform's performance if they are made available to the project.

The next iteration of this report, the "SPIRIT Platform (Second Version)" will be published in Q4/2024.

8 REFERENCES

- [1] SPIRIT Consortium, "Use Case Requirements, System Architecture and Interface Definition (First Version)," Project "Scalable Platform for Innovations on Real-time Immersive Telepresence", EC grant agreement 101070672, 2023.
- [2] SPIRIT Consortium, "Innovation Platform Enablers (First Version)," Project "Scalable Platform for Innovations on Real-time Immersive Telepresence", EC grant agreement 101070672, 2023.
- [3] SPIRIT Consortium, "SPIRIT Platform (First Version)," Project "Scalable Platform for Innovations on Real-time Immersive Telepresence", EC grant agreement 101070672, 2023.
- [4] R. Martin-Brualla, R. Pandey, S. Yang, P. Pidlypenskyi, J. Taylor, J. Valentin, S. Khamis, P. Davidson, A. Tkach and P. Lincoln, "Lookingood: Enhancing performance capture with real-time neural re-rendering," 2018.
- [5] J. Son, S. Gul, G. S. Bhullar, G. Hege, W. Morgenstern, A. Hilsmann, T. Ebner, S. Bliedung, P. Eisert and T. Schierl, "Split rendering for mixed reality: Interactive volumetric video in action," *SIGGRAPH Asia 2020 XR*, pp. 1-3, 2020.
- [6] J. Bottomley, "Building Encrypted Images for Confidential Computing," [Online]. Available: <https://blog.hansenpartnership.com/building-encrypted-images-for-confidential-computing/>. [Accessed 13 August 2023].
- [7] J. Bottomley, "Deploying Encrypted Images for Confidential Computing," [Online]. Available: <https://blog.hansenpartnership.com/deploying-encrypted-images-for-confidential-computing/>. [Accessed 13 August 2023].
- [8] A. Bentaleb, B. Taani, A. C. Begen, C. Timmerer and R. Zimmermann, "A Survey on Bitrate Adaptation Schemes for Streaming Media over HTTP," *IEEE Communications Surveys & Tutorials*, pp. 562-585, 2018.
- [9] G. Cofano, L. De Cicco, T. Zinner, A. Nguyen-Ngoc, P. Tran-Gia and S. Mascolo, "Design and Experimental Evaluation of Network-assisted Strategies for HTTP Adaptive Streaming," in *Proceedings of the 7th international conference on multimedia systems*, 2016.
- [10] B. Taraghi, M. Nguyen, H. Amirpour and C. Timmerer, "INTENSE: In-Depth Studies on Stall Events and Quality Switches and Their Impact on the Quality of Experience in HTTP Adaptive Streaming," *IEEE Access*, vol. 9, pp. 118087-118098, 2021.

- [11] M. Krivokuća, M. Koroteev and P. A. Chou, "A Volumetric Approach to Point Cloud Compression," *arXiv preprint arXiv:1810.00484*, 2018.
- [12] R. L. de Queiroz and P. A. Chou, "Compression of 3D Point Clouds Using a Region-Adaptive Hierarchical Transform," *IEEE Transactions on Image Processing*, vol. 25, no. 8, pp. 3947-3956, 2016.
- [13] Y. Huang, J. Peng, C.-C. J. Kuo and M. Gopi, "A Generic Scheme for Progressive Point Cloud Coding," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 2, pp. 440-453, 2008.
- [14] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. Cesar, P. A. Chou, R. A. Cohen and others, "Emerging MPEG Standards for Point Cloud Compression," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 1, pp. 133-148, 2018.
- [15] G. J. Sullivan, J.-R. Ohm, W.-J. Han and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," *IEEE Transactions on circuits and systems for video technology*, vol. 22, no. 12, pp. 1649-1668, 2012.
- [16] J. van der Hooft, M. Torres Vega, C. Timmerer, A. C. Begen, F. De Turck and R. Schatz, "Objective and Subjective QoE Evaluation for Adaptive Point Cloud Streaming," in *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*, 2020.
- [17] E. Alexiou, N. Yang and T. Ebrahimi, "PointXR: A Toolbox for Visualization and Subjective Evaluation of Point Clouds in Virtual Reality," in *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*, 2020.
- [18] M. K. Bekele, R. Pierdicca, E. Frontoni, E. S. Malinverni and J. Gain, "A Survey of Augmented, Virtual, and Mixed Reality for Cultural Heritage," *Journal on Computing and Cultural Heritage (JOCCH)*, vol. 11, no. 2, pp. 1-36, 2018.
- [19] S. Petrangeli, J. Famaey, M. Claeys, S. Latre and F. De Turck, "QoE-driven rate adaptation heuristic for fair adaptive video streaming," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 12, no. 2, pp. 1-24, 2015.
- [20] H. T. Tran, D. Nguyen and T. C. Thang, "An open software for bitstream-based quality prediction in adaptive video streaming," in *the 11th ACM Multimedia Systems Conference*, 225-230, 2020.
- [21] R. ITUTP, "Parametric bitstream-based quality assessment of progressive download and adaptive audiovisual streaming services over reliable transport-quality integration module," *International Telecommunication Union*, 2017.
- [22] J. Li, X. Wang, Z. Liu and Q. Li, "A QoE Model in Point Cloud Video Streaming," in *arXiv preprint arXiv:2111.02985*, 2021.

- [23] S. Vats, M. Nguyen, S. Van Damme, J. van der Hooft, M. T. Vega, T. Wauters, C. Timmerer and H. Hellwagner, "A Platform for Subjective Quality Assessment in Mixed Reality Environments," in *2023 15th International Conference on Quality of Multimedia Experience (QoMEX)*, Ghent, 2023.
- [24] E. d'Eon, B. Harrison, T. Myers and P. A. Chou, "8i Voxelized Full Bodies, version 2 – A Voxelized Point Cloud Dataset," ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) Input Document M40059/M74006, 2017.
- [25] MPEG 3DG, "Common Test Conditions for Point Cloud Compression," ISO/IEC JTC1/SC29/WG11 Coding of Moving Pictures and Audio Meeting Proceedings, 2017.
- [26] M. Nguyen, S. Vats, S. Van Damme, J. Van Der Hooft, M. T. Vega, T. Wauters, C. Timmerer and H. Hellwagner, "Impact of Quality and Distance on the Perception of Point Clouds in Mixed Reality," in *2023 15th International Conference on Quality of Multimedia Experience (QoMEX)*, 2023.
- [27] M. Nguyen, S. Vats, S. Van Damme, J. Van der Hooft, M. T. Vega, T. Wauters, F. De Turck, C. Timmerer and H. Hellwagner, "Characterization of the Quality of Experience and Immersion of Point Cloud Videos in Augmented Reality through a Subjective Study," *IEEE Access*, 2023.
- [28] J. Birch, "Efficiency of the Ishihara Test for Identifying Red-Green Colour Deficiency," *Ophthalmic and Physiological Optics*, vol. 17, no. 5, pp. 403-408, 1997.
- [29] "ITU-T P.919: Subjective test methodologies for 360° video on head-mounted displays," ITU, 2020.
- [30] L. Stahle and S. Wold, "Analysis of Variance (ANOVA)," *Chemometrics and Intelligent Laboratory Systems*, vol. 6, no. 4, pp. 259-272, 1989.
- [31] H. Abdi and L. J. Williams, "Tukey's Honestly Significant Difference (HSD) Test," *Encyclopedia of Research Design*, vol. 3, no. 1, pp. 1-5, 2010.
- [32] T. K. Kim, "T Test as a Parametric Statistic," *Korean Journal of Anesthesiology*, vol. 68, no. 6, pp. 540-546, 2015.
- [33] H. Amirpour, R. Schatz, C. Timmerer and M. Ghanbari, "On the Impact of Viewing Distance on Perceived Video Quality," in *2021 International Conference on Visual Communications and Image Processing (VCIP)*, 2021.
- [34] H. T. Tran, N. P. Ngoc, C. T. Pham, Y. J. Jung and T. C. Thang, "A Subjective Study on QoE of 360 Video for VR Communication," in *2017 IEEE 19th international workshop on multimedia signal processing (MMSP)*, 2017.

- [35] M. Nguyen, S. Vats and H. Hellwagner, “No-Reference Quality of Experience Model for Dynamic Point Clouds in Augmented Reality,” in *3rd Mile-High Video Conference*, 2024.
- [36] C. Timmerer and A. C. Begen, “A Journey Towards Fully Immersive Media Access,” in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019.
- [37] L. Wang, C. Li, W. Dai, J. Zou and H. Xiong, “Qoe-driven and tile-based adaptive streaming for point clouds,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.
- [38] M. Kowalski, Naruniec and M. Daniluk, “Livescan3D: A Fast and Inexpensive 3D Data Acquisition System for Multiple Kinect v2 Sensors,” in *IEEE International Conference on 3D Vision*, 2015.

APPENDIX A: SECURITY

A.1 ESTABLISHING THE VPN CONNECTION WITH THE LAB INFRA-STRUCTURE

The lab infrastructure is protected by a Cisco ASA Firewall and VPN endpoint. This requires the use of the Cisco AnyConnect protocol to establish a VPN connection.

If you want to connect using the open source openconnect software issue the following command:

```
openconnect          caramel.mine.bz          --servercert          "pin-  
sha256:EpPc6Du8rBMM3B3KRe3QT1Sk+rQ9zbL4KV1/zFAVnHE=" -u cloudtest --  
authgroup cloud_vpn
```

Note on openconnect: We have found that the latest openconnect 9 software has a bug when connecting to our ASA with latest software version. Please use openconnect 8.20 to work around that problem.

Note 2: We are currently using a self-signed certificate for our Cisco ASA VPN system. The “servercert” option of the openconnect command contains the fingerprint of this certificate.

Request a username (instead of the sample “cloudtest”) and password from the lab admin.

Alternatively, if you want to use the graphical Cisco Secure Client software (available for Linux, MacOS, and Windows) open the ASA home page in web browser with the URL

➔ <https://caramel.mine.bz>

Since the certificate is self-signed, you must accept the untrusted web site.

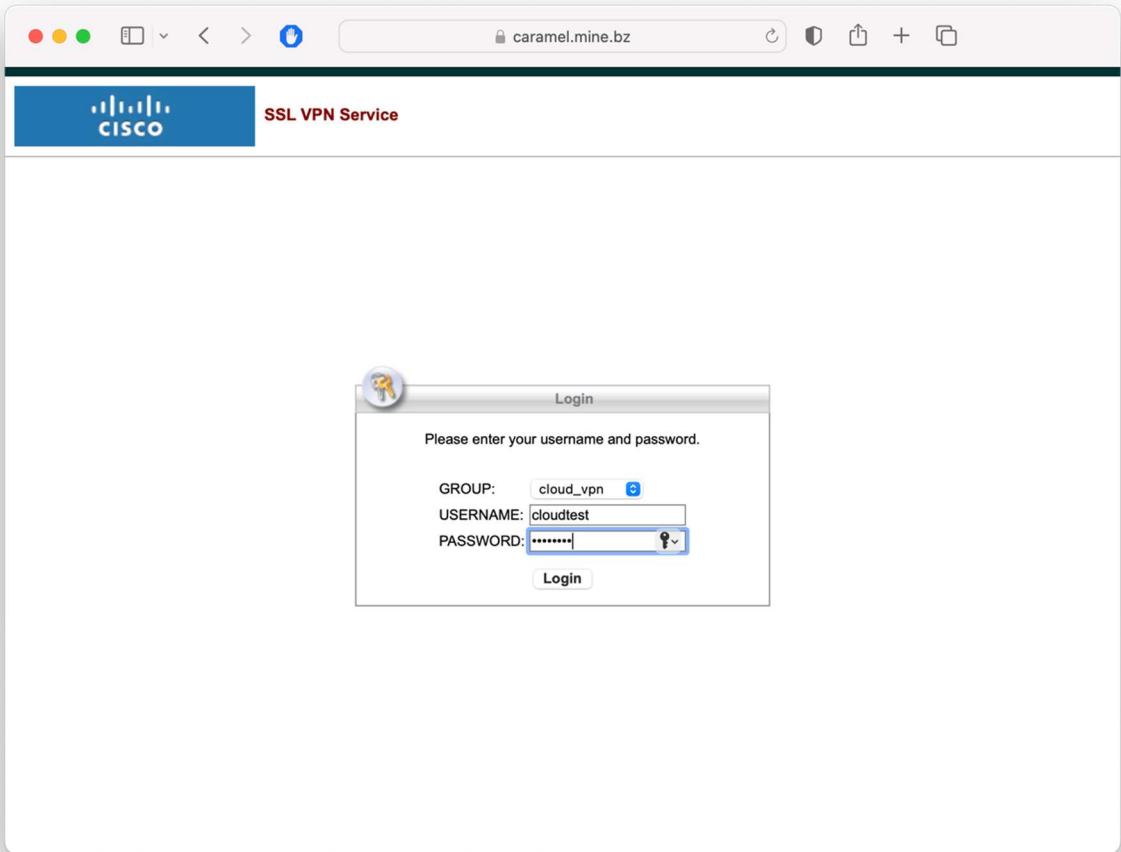


FIGURE 44: CISCO ASA LOGIN SCREEN

Figure 44 shows the Cisco ASA login screen: Login using your credentials as communicated by us.

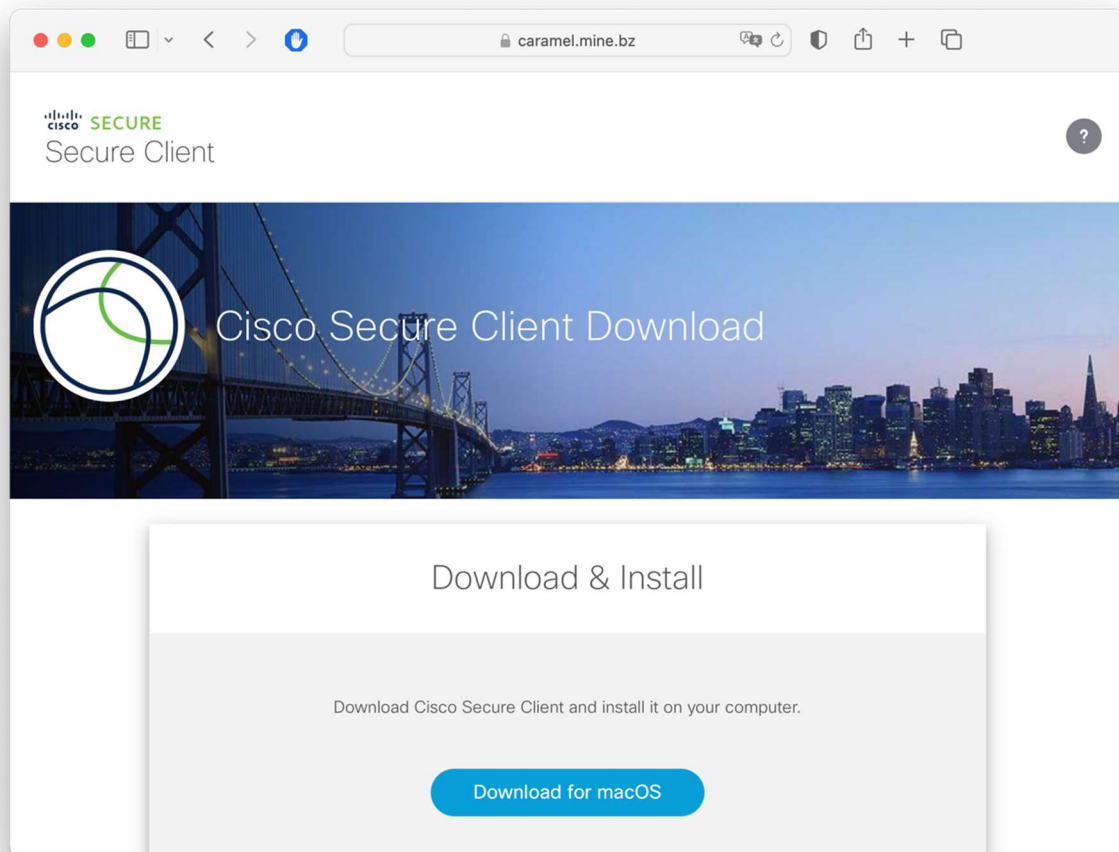


FIGURE 45: CISCO SECURE CLIENT DOWNLOAD PAGE

After login, you are brought to the Cisco Secure Client download screen (see Figure 45). Download the Secure Client for your operating system and install. Using the installed Cisco Client, you can establish the VPN connection (Figure 46).

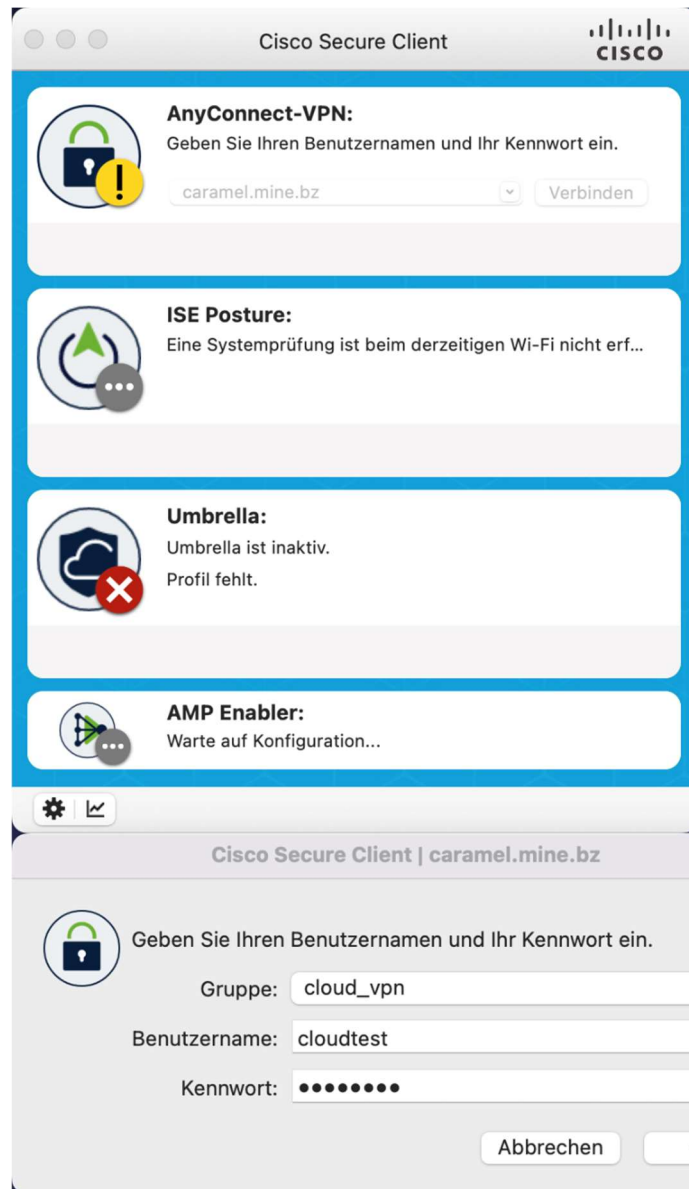


FIGURE 46: VPN CONNECTION WITH CISCO SECURE CLIENT

Notes on the Cisco AnyConnect VPN service in our lab environment:

- The VPN is required for download and upload to our NAS system.
- The VPN is also required for remote management of guest VMs. For this use case it is probably easiest to use the openconnect command line tool for automatic VPN setup.
- You will receive a dynamic IP address in the 10.0.12.0/24 subnet.
- All other traffic is routed normally to the Internet.
- The Cisco AnyConnect protocol also works behind firewalls through web proxies.
- UserIDs and passwords for VPN, NAS, and remote management (generally all the same if not otherwise requested) must be requested from a lab admin.
- We are able to open ports on our outside firewall interface (both IPv4 and IPv6) for application-specific purposes.

A.2 ACCESS NAS DEVICE FOR SOFTWARE DOWNLOAD AND VM UPLOAD

The NAS (network attached storage) device is accessible over the VPN tunnel under the URL:

➔ <https://10.0.12.20:444/cgi-bin/>

Login with your lab credentials (user ID and password).

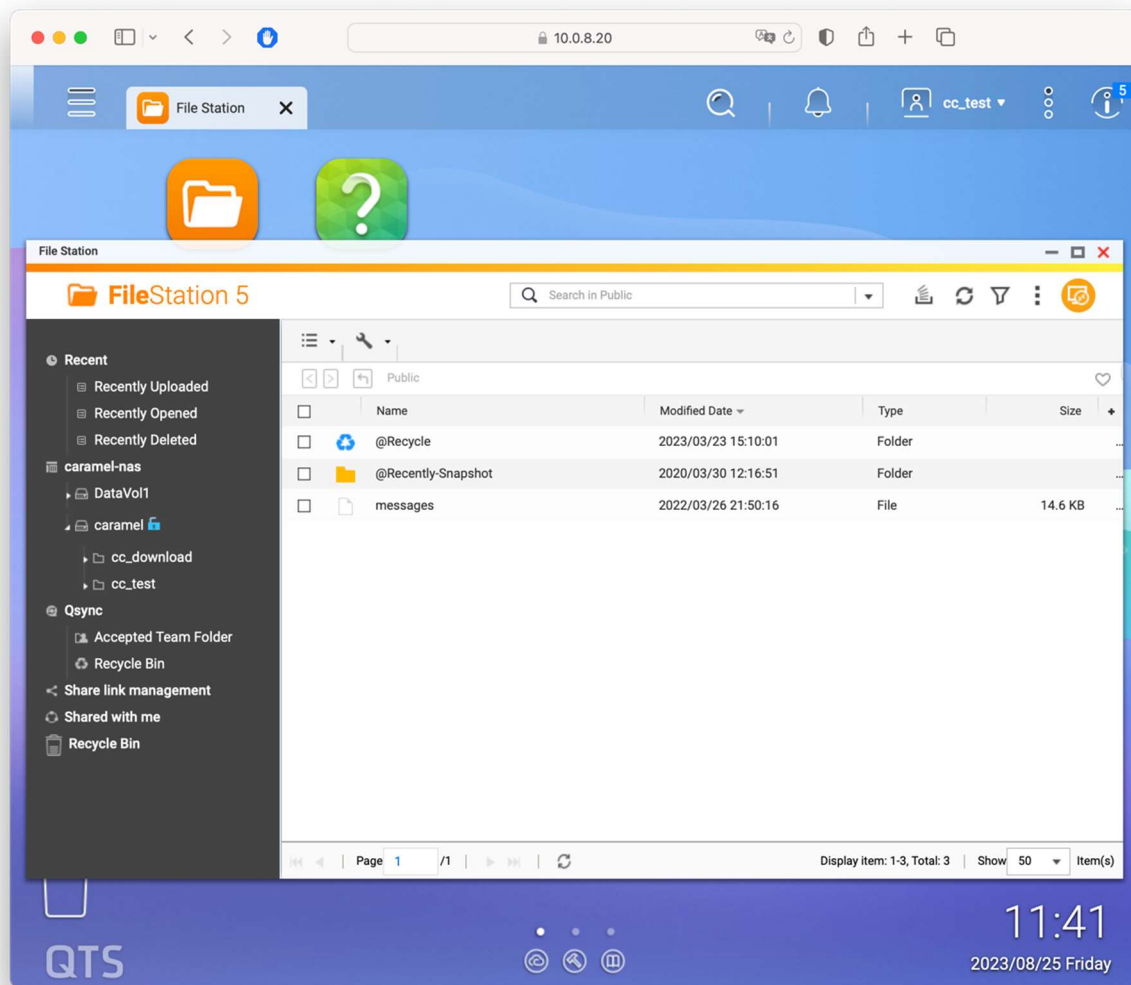


FIGURE 47: NAS GUI WITH FILESTATION APPLICATION

Figure 47 shows the NAS GUI in the web browser after clicking on the “FileStation” icon. Logged in with the sample user “cc_test”, the following two folders are of interest:

- cc_test (will be different for your user): This folder is writable. Upload your completed and encrypted guest VM images here.
- cc_download: This folder contains the tool downloads:
 - Readme.md: (optional) Markdown file with latest information on using the tool set that may not be in the latest version of this document.
 - remote-management.tar: TAR file of the exported Docker image
 - remote-management-workdir.tar: Working directory to be used in conjunction with the Docker image
 - cc-setup-vm.ova: OVF/OVA file with the tool VM for import into your hypervisor.
 - cc-setup-workdir.tar.gz: TAR file with the workdir directory structure for setup tool VM.

A.3 USING THE SUPPLIED VM OVA FILE

Partners will be given the Guest VM Setup tool VM's image in OVA (`cc-setup-vm.ova`) format to import in their VMM (Virtual Machine Manager) of choice. The following steps show how to import the OVA file into the Virtual VMM on Linux.

Note that these steps have been successful on our test system. The specific steps might vary depending on your host operating system or hypervisor software. Please contact our support team if there are any difficulties installing or accessing the tool VM.

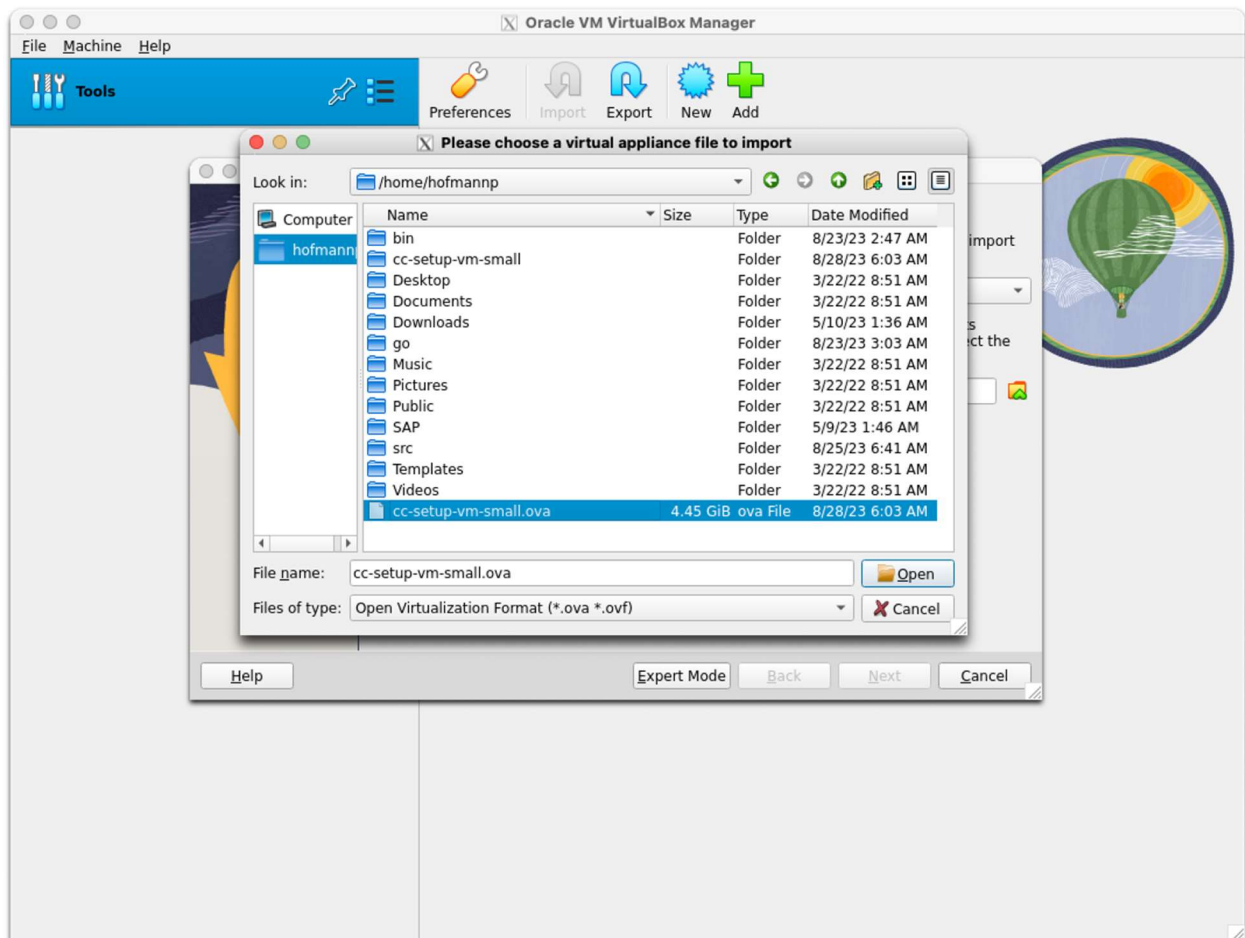


FIGURE 48: IMPORT OF OVA FILE

Figure 48 shows how to select and import the `cc-setup-vm.ova` file into VirtualBox.

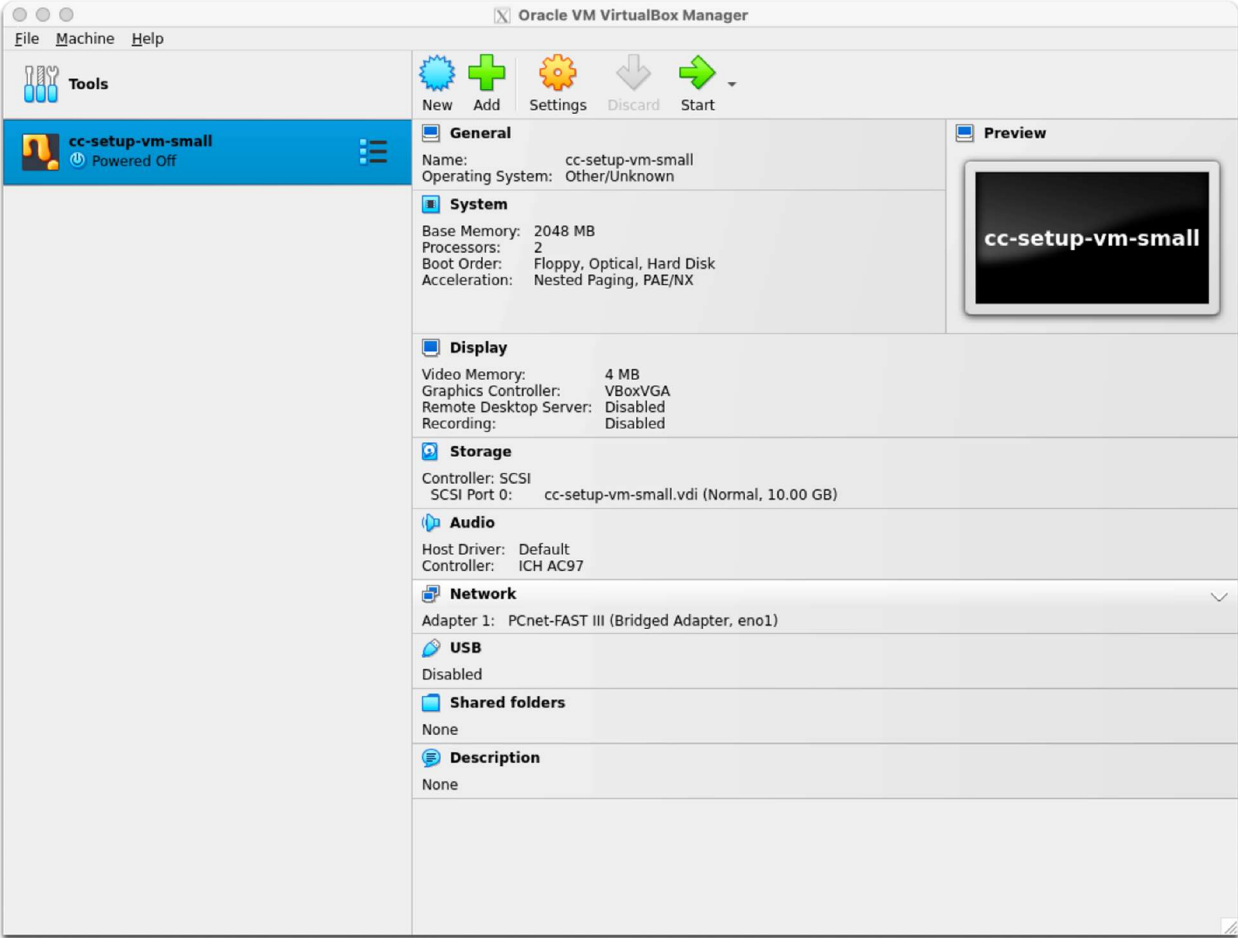


FIGURE 49: RESULT OF OVA IMPORT

Figure 49 shows the resulting screen in VirtualBox after the import.

Then “Tools” on the left and create a NAT network for VirtualBox (see Figure 50).

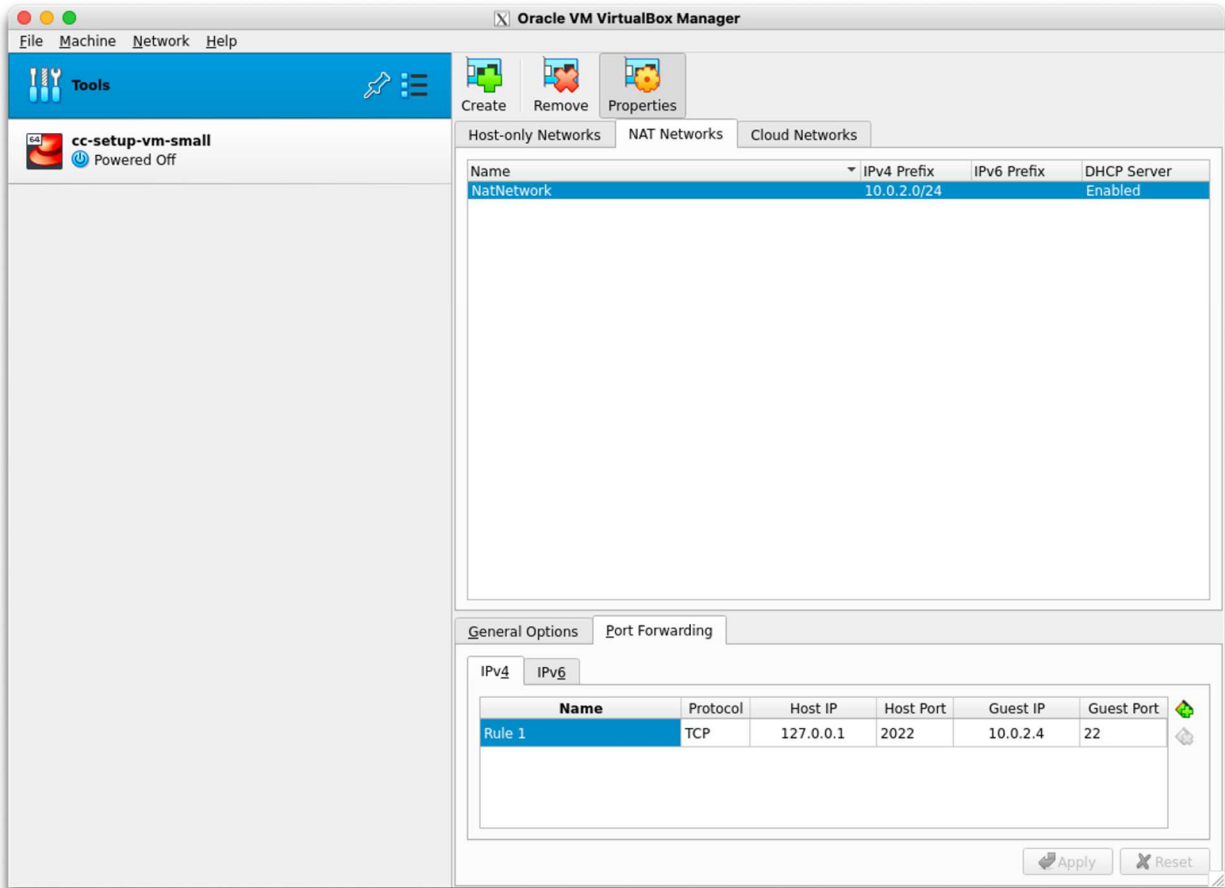


FIGURE 50: NAT NETWORK IN VIRTUALBOX

The NAT network should have the following parameters:

- Prefix 10.0.2.0/24
- DHCP enabled
- A port forwarding from host (127.0.0.1) port 2022 to port 22 on the guest system (10.0.2.4, IP address might vary in your setup – look into the VM once booted).

Now select the VM “cc-setup-vm-small” on the left and press “Settings”.

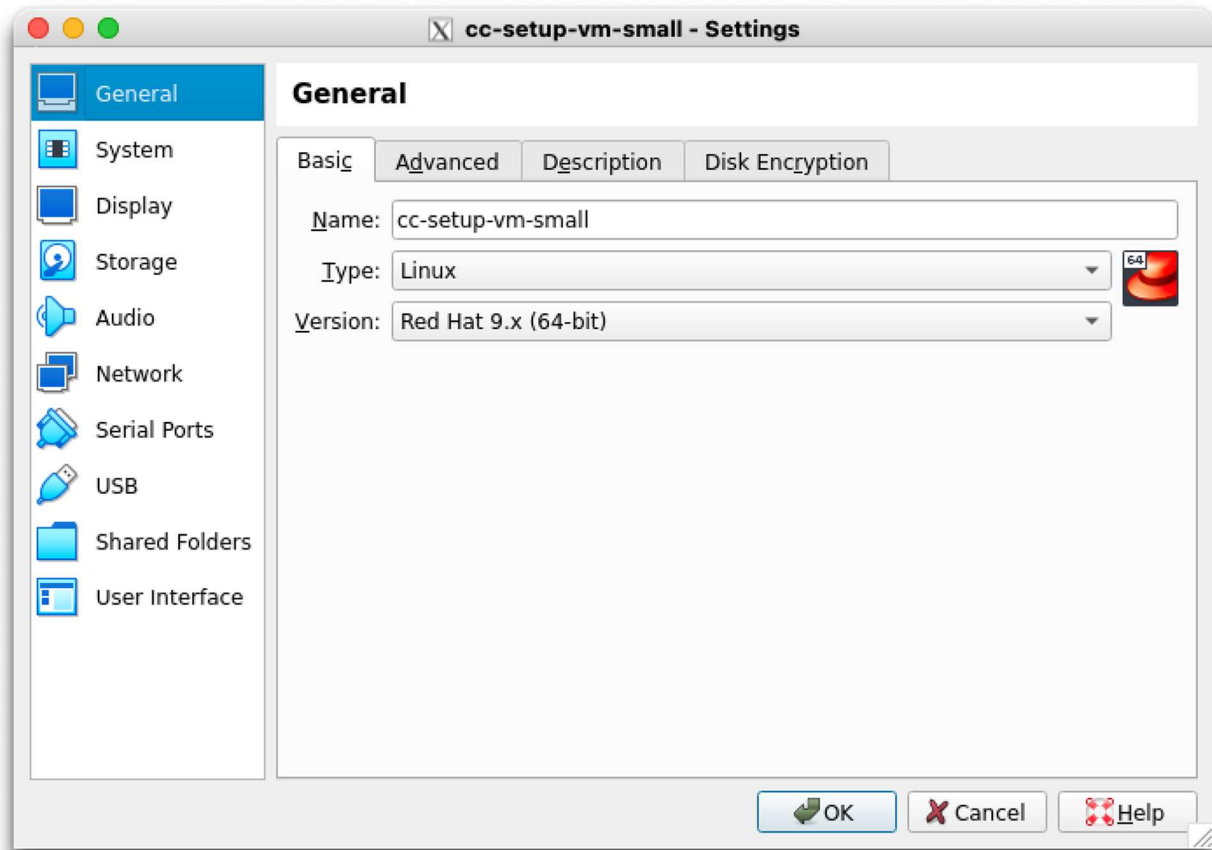


FIGURE 51: CHANGE VM OS AND VERSION

Figure 51 shows how to change the OS to “Linux” and the OS version to “Red Hat 9.x (64-bit)”. This is necessary because our tool VM is a 64-bit Centos Linux system.

Now go to the Storage tab to create a large disk for the workdir.

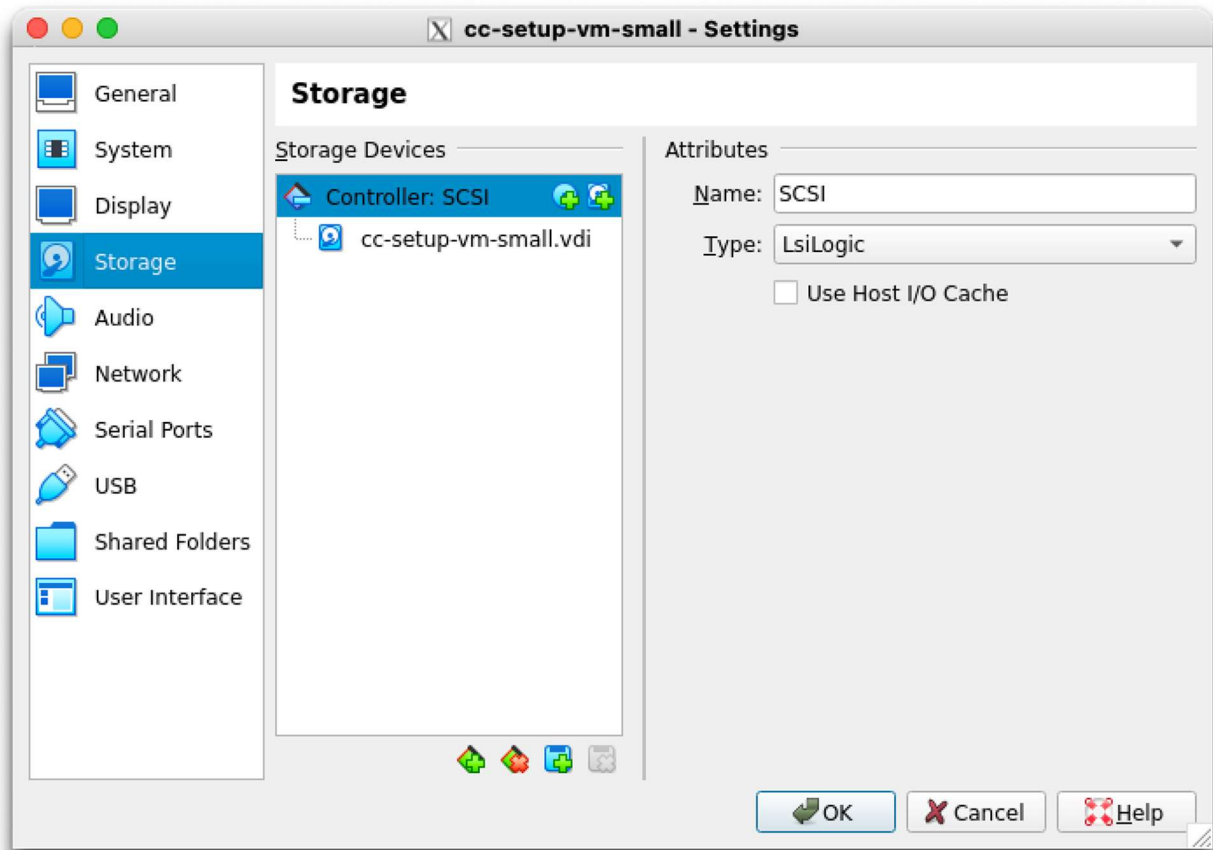


FIGURE 52: STORAGE TAB IN VM SETTINGS

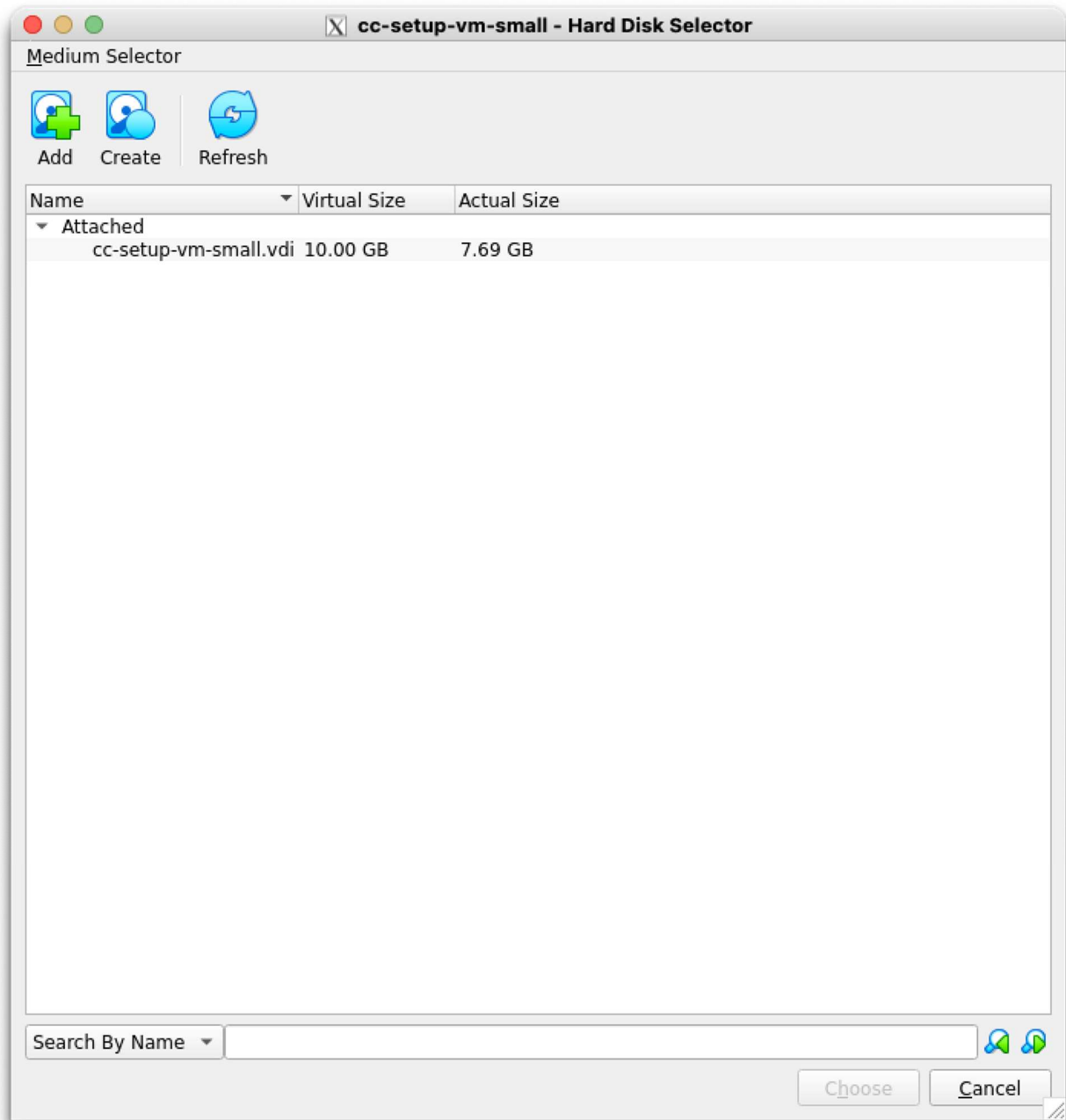


FIGURE 53: HARD DISK SELECTION IN VIRTUALBOX

Figure 53 shows the hard disk selection dialog after pressing “Add disk”. Here, press “Create” to create a new disk that is later used as the working directory for creating new guest images.



FIGURE 54: VIRTUAL DISK FILE TYPE DIALOG



FIGURE 55: VIRTUAL DISK ALLOCATION DIALOG

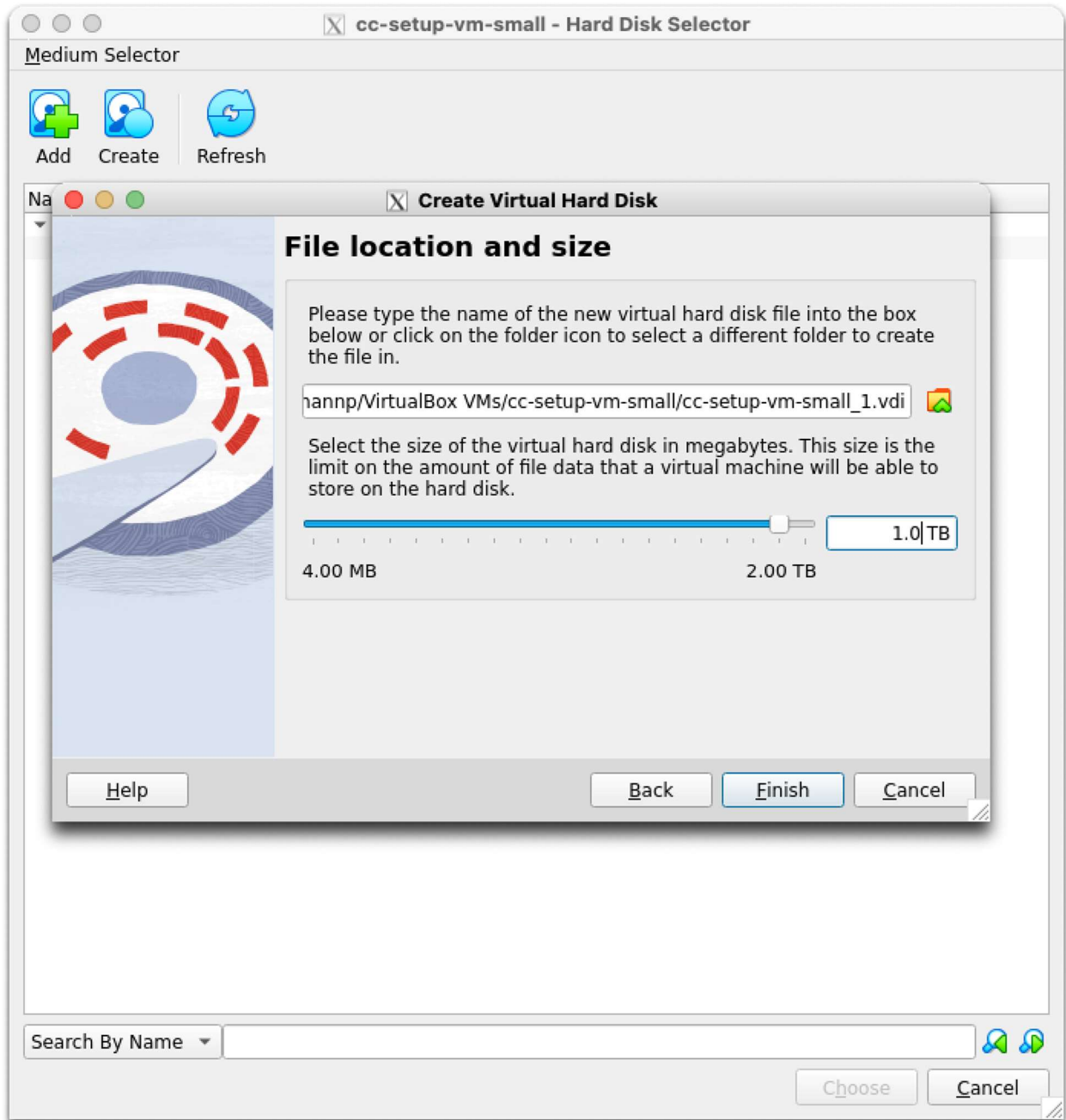


FIGURE 56: VIRTUAL DISK SIZE DIALOG

Figure 54, Figure 55, and Figure 56 show the different steps for the new virtual disk creation:

- Use the type „VDI“
- Do not pre-allocate all storage (saves disk space by using sparse files)
- Define a size large enough to hold the templates as well as any number of guest images you might want to create (in this case 1 TB).

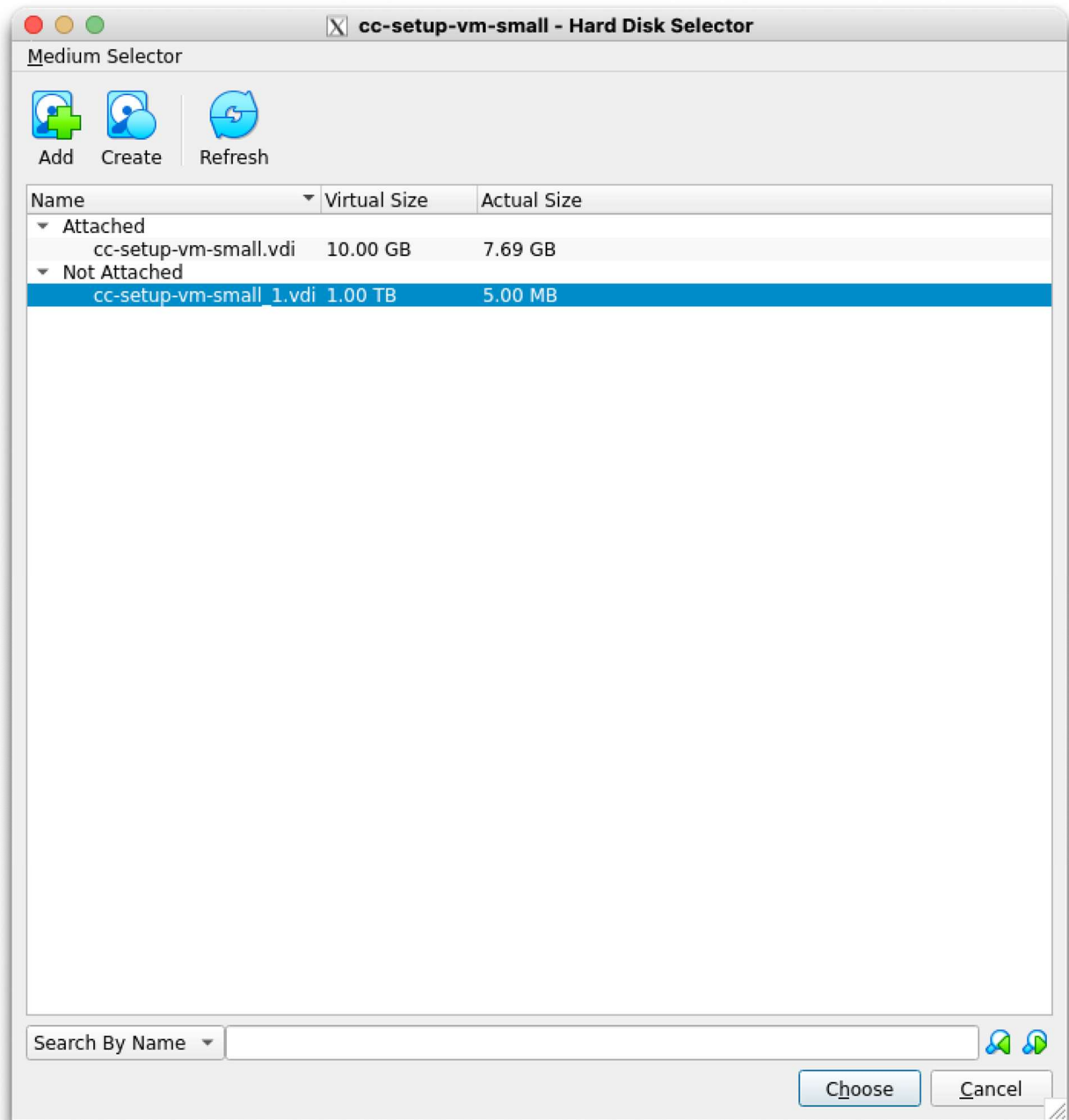


FIGURE 57: RESULT OF VIRTUAL DISK CREATION

Figure 57 shows the result of the disk creation: A new 1TB disk has been created, that only uses 5 MB of disk space at the moment.

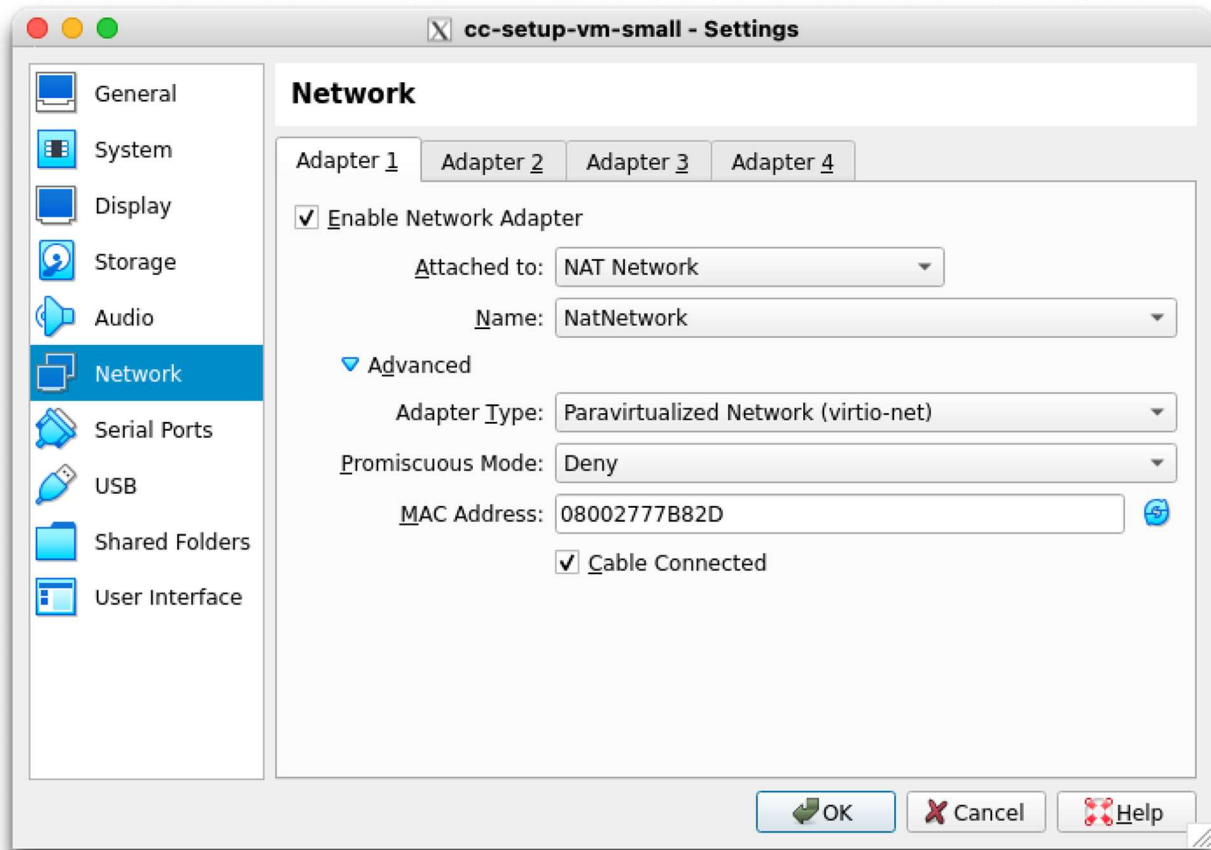


FIGURE 58: VIRTUALBOX NETWORK CONFIGURATION

Figure 58 show the final configuration of the network interface in the VirtualBox appliance:

- Select the NAT network „NatNetwork“ define previously.
- Choose adapter type “virtio-net”
- Select “Cable Connected”

Now startup the virtual machine. Because we have created a port forwarding, you can log into the virtual machine like this:

```
ssh -p2022 -Y user@127.0.0.1
su -
```

Note that the password for user and root is “udel1718”.

As root, check that the /dev/sdb is the new 1TB virtual disk:

```
$ fdisk -l /dev/sdb
```

```
Disk /dev/sdb: 1 TiB, 1099511627776 bytes, 2147483648 sectors
```

```
Disk model: VBOX HARDDISK
```

```
Units: sectors of 1 * 512 = 512 bytes
```

```
Sector size (logical/physical): 512 bytes / 512 bytes
```

```
I/O size (minimum/optimal): 512 bytes / 512 bytes
```


Then, create a file system on this disk:

```
$ fdisk /dev/sdb
```

```
Welcome to fdisk (util-linux 2.37.4).
```

```
Changes will remain in memory only, until you decide to write them.
```

```
Be careful before using the write command.
```

```
Device does not contain a recognized partition table.
```

```
Created a new DOS disklabel with disk identifier 0xe9d0e0fa.
```

```
Command (m for help): n
```

```
Partition type
```

```
  p   primary (0 primary, 0 extended, 4 free)
```

```
  e   extended (container for logical partitions)
```

```
Select (default p): p
```

```
Partition number (1-4, default 1):
```

```
First sector (2048-2147483647, default 2048):
```

```
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-2147483647, default 2147483647):
```

```
Created a new partition 1 of type 'Linux' and of size 1024 GiB.
```

```
Command (m for help): w
```

```
The partition table has been altered.
```

```
Calling ioctl() to re-read partition table.
```

```
Syncing disks.
```

```
$ mkfs.xfs /dev/sdb1
```

```
meta-data=/dev/sdb1          isize=512    agcount=4, agsize=67108800 blks
      =                       sectsz=512   attr=2, projid32bit=1
      =                       crc=1          finobt=1, sparse=1, rmapbt=0
      =                       reflink=1     bigtime=1 inobtcount=1 nrext64=0
data      =                   bsize=4096   blocks=268435200, imaxpct=25
      =                       sunit=0       swidth=0 blks
naming   =version 2          bsize=4096   ascii-ci=0, ftype=1
log      =internal log     bsize=4096   blocks=131071, version=2
      =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none              extsz=4096   blocks=0, rtextents=0
```

```
$ mkdir /data
```

Add the new filesystem to `/etc/fstab`:

```
$ vi /etc/fstab
#
# /etc/fstab
# Created by anaconda on Mon Aug 28 09:24:49 2023
#
# Accessible filesystems, by reference, are maintained under '/dev/disk/'.
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info.
#
# After editing this file, run 'systemctl daemon-reload' to update systemd
# units generated from this file.
#
/dev/sda1 /                xfs     defaults    0 0
/dev/sdb1 /data                   xfs     defaults    0 0
```

Mount the disk:

```
$ /bin/mount -a
mount: (hint) your fstab has been modified, but systemd still uses
        the old version; use 'systemctl daemon-reload' to reload.
$ systemctl daemon-reload
```

Unpack the working dir tar (obtained from the NAS):

```
$ cd /data
$ tar xvf cc-setup-workdir.tar.gz
```

And link the the new working dir to `/home/user/workdir`:

```
$ cd /home/user/
$ rm -rf workdir
$ ln -s /data/cc-setup-workdir workdir
```

The system is now ready to create guest images as described in main part of this document.

Note: Alternatively, guest owner can request a KVM-compatible QCOW2 image from us that can directly be used with the Linux KVM hypervisor. This file has already 1 TB of free space on the home partition.

A.4 END-TO-END EXAMPLE

In this section we describe a complete end-to-end procedure for obtaining the necessary tools, preparing a Centos 9-based guest VM, and finally deploying it on our lab system. This section is meant to be a step-for-step guide for project partners for preparing their own workloads as protected images to run on our platform.

Obtaining and Installing the Software

Download the following files from the NAS device in our lab (from the cc_download shared folder, refer to the appendix):

- remote-management.tar: TAR file of the exported Docker image
- remote-management-workdir.tar: Working directory to be used in conjunction with the Docker image
- cc-setup-vm.ova: OVF/OVA file with the tool VM for import into your hypervisor.
- cc-setup-workdir.tar: Tool VM workdir with the prepared Centos 9 and Debian 11 templates (rename/move the included directory to /home/user/workdir).

Import the Docker image into your local repository and unpack the workdir for the Docker image:

```
docker load --input remote-management.tar
tar xvf remote-management-workdir.tar
```

The import of the OVF/OVA file cc-setup-vm.ova is described in the appendix.

Preparing the Guest VM

The guest VM must be prepared in the tool VM provided by us. Start the guest VM in your hypervisor (eg. VirtualBox) and login as “root” with password “udel1718”. Alternatively, you can use the user “user” with password “udel1718”, but the commands must be run as root because they require special privileges.

Prepare your guest VM from the Centos 9 template with this command (please choose your own recovery password):

```
prepare --centos centos-test Luci-0815-deli
```

The prepare script outputs the following:

```
Verifications...
Verifications OK
Password management...
33+0 records in
33+0 records out
33 bytes copied, 0.00022163 s, 149 kB/s
Password management OK
Partition is /dev/nbd9
Connected qemu-nbd device
Encrypting partition p2...
Added user password
Added high-entropy password
User password and high entropy password correctly added, deleting standard password...
AMD files...
```

```

PDH EP384 D256 60f50e13747224658c3d84bebc21a2e888525649185f62ea19d314d7bf7eb72b
  ↳ PEK EP384 E256 0a8e2bfd971103857c117e2b3b37bf8865a75441d945cbd9e46d7db764c5ef20
    •↳ OCA EP384 E256
aed15dc56800252cc82114964cac98f47f42c068f19feaa189d14a2321800b06
  ↳ CEK EP384 E256
537e69f2709c213aa69be03ae8be5307cf7963708de56a7a8658aed29117ab07
  ↳ ASK R4096 R384
95cba79ba3c77daea79f741bade8156a50b1c59f6d6fda104d16dd264729f5ee8989522f3711fc7c847
19921ceb31bc0
    •↳ ARK R4096 R384
569da618dfe64015c343db6d975e77b72fdeacd16edd02d9d09b889b8f0f1d91ffa5dfbd86f7ac574a1
a7883b7a1e737

```

• = self signed, ↳ = signs, •/ = invalid self sign, ↳/ = invalid signs

AMD files OK

Disabling Debian LVM...

umount: /dev/mapper/debian--vg-root: no mount point specified.

Unmounting partitions...

umount: /dev/mapper/encrypted-image: no mount point specified.

umount: /dev/mapper/encrypted-boot: no mount point specified.

Device encrypted-image is not active.

Device encrypted-boot is not active.

Disconnecting nbd devices...

/dev/nbd0 disconnected

/dev/nbd1 disconnected

/dev/nbd10 disconnected

/dev/nbd11 disconnected

/dev/nbd12 disconnected

/dev/nbd13 disconnected

/dev/nbd14 disconnected

/dev/nbd15 disconnected

/dev/nbd2 disconnected

/dev/nbd3 disconnected

/dev/nbd4 disconnected

/dev/nbd5 disconnected

/dev/nbd6 disconnected

/dev/nbd7 disconnected

/dev/nbd8 disconnected

/dev/nbd9 disconnected

qemu-nbd: Cannot open /dev/nbd9p1: No such file or directory

qemu-nbd: Cannot open /dev/nbd9p2: No such file or directory

Prepare : DONE

Please check the output for any errors or problems and contact our support team if there are any issues. Sometimes the prepare process only terminates successfully after trying a second time.

As a result of this process, the transfer directory contains a file `centos_centos-test.qcow2`:

```
[root@localhost workdir]# ls -lsk transfer/
total 51350352
51350352 -rw-r--r--. 1 root root 1099679662080 Aug 25 08:28 centos_centos-test.qcow2
```

Note that this is a sparse file, ie. while the file size is shown as 1099679662080 (around 1TB), the actual number of 1024 sectors is only 51350352.

Additionally, this process created two key files in the `workdir/keys` subdirectory:

- `recovery-password_centos-test.txt`: This file contains the low-entropy key (ie. short password) that was given on the command line: “Luci-0815-deli”.
- `high-entropy-password_centos-test.txt`: This file contains a high-entropy (ie. long password) that was generated during the prepare process. Because it is a high-entropy password, it was added with few PBKDF2 iterations in order to speed up the boot process of the guest VM later on.

Note: these two passwords must never be divulged to the cloud operator (our lab admin). If, in the course of debugging the process, we gain insight into these passwords, the guest owner should re-create the guest VM image with different passwords in private afterwards.

In a next step the guest owner might want to perform a number of customization steps, such as:

- Updating the underlying Linux distribution using `apt` or `dnf`. **Warning: a blanket “dnf update” or “apt upgrade” might lead to a re-build of the initramfs filesystem which breaks the boot process because the initramfs is built on a different kernel than the final guest VM will boot. Any dnf or apt actions that might lead to a rebuild of the initramfs must be avoided. If unsure, it is better to postpone any dnf/apt commands until the VM is running in the lab environment.**
- Installing additional software from the distribution or other repositories.
- Installation of own binaries or data needed for the application.
- Re-configuration of the Linux operating system.

Issue

```
mount centos-test
```

to invoke a `chroot` environment with all necessary directories of the guest VM mounted. This environment can be used almost like a VM actually running on the target system. For example, try to install a sample hello world application:

```
[root@localhost /]# echo '#!/bin/bash' > /usr/local/bin/hello_world.sh
[root@localhost /]# echo 'echo "Hello World!"' >> /usr/local/bin/hello_world.sh
[root@localhost /]# chmod +x /usr/local/bin/hello_world.sh
```

```
[root@localhost ~]#
```

Now you should perform further steps to personalize the guest VM:

- Change the root and user password from the default “Udel1718” to a secure password only known to you (“passwd root” and “passwd user” commands).
- Regenerate the host keys and make a copy of the public keys so that you have unique, trusted host keys for later SSH login and administration (see the appendix for details for Debian and Centos).

Follow this procedure to re-generate the host keys:

Delete the old host keys:

```
rm -f /etc/ssh/ssh_host*
```

Regenerate the host keys:

```
ssh-keygen -f /etc/ssh/ssh_host_rsa_key -N '' -q -t rsa
```

```
ssh-keygen -f /etc/ssh/ssh_host_ecdsa_key -N '' -q -t ecdsa
```

```
ssh-keygen -f /etc/ssh/ssh_host_ed25519_key -N '' -q -t ed25519
```

Make a copy of the .pub files and enter them into your local .ssh/known_hosts file.

Exit the chroot environment by typing “exit” or CTRL-D. The mounted directories from the guest image must be unmounted using:

```
umount-images
```

Now the guest image is ready for transfer to the lab environment. In order to preserve the sparse file properties, use TAR to compress the image before transfer:

```
cd ~/workdir/transfer
```

```
tar -S -zcvf centos_centos-test.tar.gz centos_centos-test.qcow2
```

The resulting TAR file should then be uploaded to the designated directory on the lab NAS system (refer to the appendix for details).

As a final step, the launch bundle for the guest VM must be prepared and some parameters be transmitted to the host system. This requires a working VPN connection to our lab environment.

Now prepare a new subdirectory for our new VM centos_centos-test:

```
mkdir workdir/centos_centos-test
```

Then create the launch bundle and send it to our host system:

```
$ docker run -v /home/hofmannp/bare_metal_cc/workdir:/workdir:z remote-management create-launch-bundle /workdir/centos_centos-test
```

```
Writing to file: /workdir/centos_centos-test/godh.cert
```

```
Writing to file: /workdir/centos_centos-test/tmpTk.bin
```

```
Writing to file: /workdir/centos_centos-test/launch_blob.bin
```



Command Successful

```
16+0 records in
16+0 records out
16 bytes copied, 0.000166162 s, 96.3 kB/s
16+0 records in
16+0 records out
16 bytes copied, 0.000161821 s, 98.9 kB/s
```

```
$ docker run -v /home/hofmannp/bare_metal_cc/workdir:/workdir:z remote-management
send http://10.0.12.96:5000 centos_centos-test user /workdir/passwords/user_pwd.txt
/workdir/centos_centos-test
```

Files were successfully sent

```
[hofmannp@r370-2 bare_metal_cc]$
```

Note: This command requires that the user “user” (sample user name used for documentation purposes) is registered for the guest owner and the password has been written into the `workdir/passwords/user_pwd.txt` file. Of course, instead of “user” you would need to use your VPN user ID and password for this.

Note on SE Linux: If the host system is Linux and implements SE Linux mandatory access control, the Docker container might not be able to access the files in the host’s working directory. You might need to issue commands like “`chcon unconfined_u:object_r:container_file_t:s0 workdir/passwords/user_pwd.txt`” to adapt SE Linux file labels.

Now the lab admin must be notified about the upload of the TAR.GZ file and the sending of the certificates. After confirmation, the VM can be started using the remote management commands described in the next section.

Running the Guest VM

The following command starts the remote VM in paused mode:

```
$ docker run -ti --network="host" -v /home/hofmannp/bare_metal_cc/workdir:/workdir:z
remote-management start http://10.0.12.96:5000 centos_centos-test user /workdir/pass-
words/user_pwd.txt
```

VM `centos_centos-test` was successfully started in paused mode

Now that the VM is started in paused mode, we must perform the remote attestation/measurement and inject the high entropy disk key/password:

```
$ docker run -v /home/hofmannp/bare_metal_cc/workdir:/workdir:z remote-management
launch /workdir/centos_centos-test/high-entropy-password_centos-test.txt /work-
dir/centos_centos-test 10.0.12.96:5551
```

```
SEV query found API=1.53 build=5 policy=3
```

Getting Launch Measurement

```
Measure:          9a311c81d8e8078485f005c207a7519d2e971c9288dd906658dfc25269d1fc5a
```



```
Measure (b64): b'mjEcgdjoB4SF8AXCB6dRnS6XHJKI3ZBmWN/CUmnR/Fo='  
should be:      9a311c81d8e8078485f005c207a7519d2e971c9288dd906658dfc25269d1fc5a
```

Measurement matches, Injecting Secret

Secret Injection Successful, starting VM

Since the measurements match, the VM is started. Note the QMP protocol host and port “10.0.12.96:5551”. While the host IP address is fixed in our test environment, the QMP port 5551 must be defined for each VM. The port number is allocated by the lab admin during setup and communicated via email to the guest owner.

When looking at the GUI screen in our test environment, we can see that the hello word application is present (Figure 59):

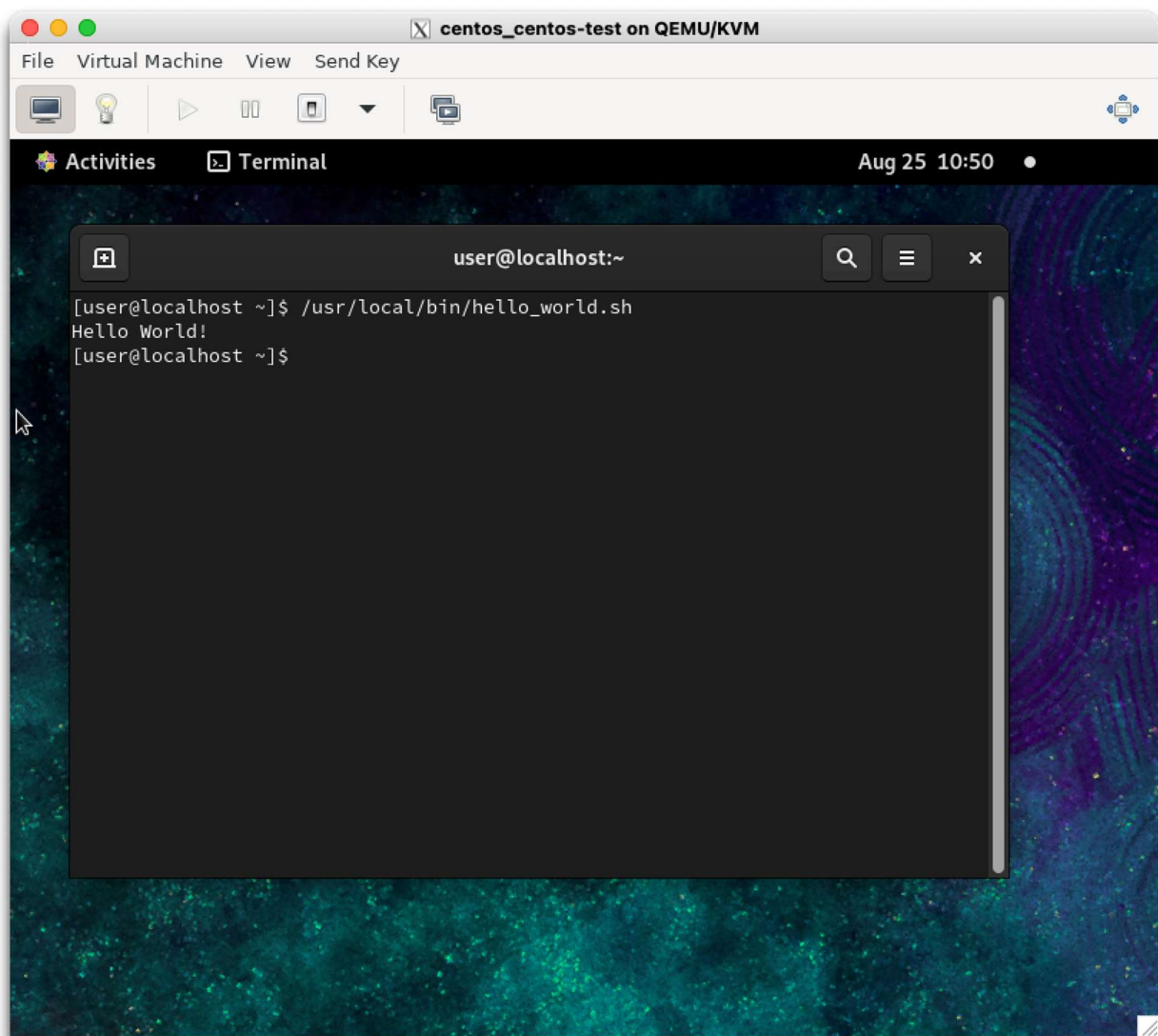


FIGURE 59: HELLO WORD APPLICATION IN THE TRUSTED GUEST VM

In order to stop the VM issue the shutdown and destroy commands:

```
$ docker run -v /home/hofmannp/bare_metal_cc/workdir:/workdir:z remote-management shutdown http://10.0.12.96:5000 centos_centos-test user /workdir/passwords/user_pwd.txt
```

Command was sent successfully but VM encountered a problem. Current VM status is {"status":"running"}

```
$ sleep 30
```

```
$ docker run -v /home/hofmannp/bare_metal_cc/workdir:/workdir:z remote-management destroy http://10.0.12.96:5000 centos_centos-test user /workdir/passwords/user_pwd.txt
```

VM centos_centos-test was successfully shutdown

Security Checklist

To make sure that your VM really is really protected when running in the host environment, make sure:

- To not divulge the recovery and high-entropy passwords to anyone (not even the DT support team) and store them on your guest system in a safe location.
- Re-generate the SSH host keys and import them into your local SSH client configuration.
- Do not connect to a remote VM if the host keys seem to have changed!
- Change the default user and root passwords to something long and unique. The default passwords are mentioned in this document are not secure!

For added security, you could check the integrity of the template VMs by comparing files to the original Centos 9 and Debian distributions. You could also re-compile the OVMF/grub bundle and provide your own OVMF image to us.

A.5 FORMAL DEFINITION OF REMOTE MANAGEMENT WEB SERVICE (YAML)

For reference purposes we reproduce the YAML file for remote management REST service here:

```
openapi: 3.0.0
info:
  title: API for remote VM management
  version: 1.0.0
paths:
  /vm/start:
    post:
      summary: Start a VM in pause mode
      operationId: "vm.start"
```

```
requestBody:
  required: true
  content:
    application/json:
      schema:
        type: object
        properties:
          vm_name:
            type: string
          userid:
            type: string
          password:
            type: string
        required:
          - vm_name
          - userid
          - password
responses:
  '200':
    description: OK
  '500':
    description: Internal server error
/vm/destroy:
  post:
    summary: Destroy VM
    operationId: "vm.destroy"
    requestBody:
      required: true
      content:
        application/json:
          schema:
            type: object
            properties:
              vm_name:
                type: string
              userid:
                type: string
              password:
                type: string
            required:
              - vm_name
              - userid
              - password
```

```
responses:
  '200':
    description: OK
  '500':
    description: Internal server error
/vm/shutdown:
post:
  summary: Shutdown VM
  operationId: "vm.shutdown"
  requestBody:
    required: true
    content:
      application/json:
        schema:
          type: object
          properties:
            vm_name:
              type: string
            userid:
              type: string
            password:
              type: string
          required:
            - vm_name
            - userid
            - password
responses:
  '200':
    description: OK
  '500':
    description: Internal server error
/vm/status:
post:
  summary: Return status of the host
  operationId: "vm.status"
  requestBody:
    required: true
    content:
      application/json:
        schema:
          type: object
          properties:
            vm_name:
```

```

        type: string
    userid:
        type: string
    password:
        type: string
    required:
        - vm_name
        - userid
        - password
responses:
    '200':
        description: OK
    '500':
        description: Internal server error

```

A.6 UPDATING THE TEMPLATE FILES

The Debian 11 and Centos 9 templates must be regularly updated to fix bugs and security problems. This activity needs to be carried out by the provider of the confidential computing infrastructure on a regular basis, so this section is not relevant to partners wanting to use the provided templates to create guest VMs.

Before the update, the template images should be saved to have a fallback position if there are problems with the updated images:

```
tar --hole-detection=seek -S -cvf template-backup.tar centos9-encrypted-template.qcow2 debian11-encrypted-template.qcow2
```

This command preserves the sparse file properties of the templates. The resulting `template-backup.tar` can also be used to transfer the updated images later to the tool VM.

After saving the images, the Debian 11 and Centos 9 images should be started in turn.

On Debian 11, as root, the following commands update the packages:

```
apt update
```

```
apt upgrade
```

For Debian 11, the following lines have been added to `/etc/apt/apt.conf.d/50unattended-upgrades` to avoid kernel upgrades:

```
Unattended-Upgrade::Package-Blacklist {
    "linux-generic";
    "linux-image-generic";
    "linux-headers-generic";
};
```

On Centos 9, as root, the following command updates the packages:

```
dnf update
```

For Centos 9, the following line has been added to `/etc/dnf/dnf.conf` to avoid kernel upgrades:

```
exclude=kernel* redhat-release* kmod-kvdo
```

After both templates are checked to still work with measured start and basic OS operation is possible, re-create the TAR file as outlined above and use it to update the templates in the tool VM.

Note on Kernel upgrades: Kernel upgrades requires manual changes to the `grub.cfg` because both templates have been modified to be “Confidential Computing-enlightened”. The procedures for kernel upgrades will be documented in later versions of this document.

In the tool VM, the TAR file must be unpacked. The template QCOW2 files must be renamed to `centos9-template.qcow2` and `debian-template.qcow2`, respectively. Afterwards, issue the following commands as root to reclaim space for optimal download size of the tool VM:

```
rm -f /home/user/workdir/keys/* /home/user/workdir/transfer/*
```

```
fstrim -av
```

The tool VM must then be exported as OVF/OVA file using the following command:

```
$ sudo /home/hofmannp/go/bin/ovf-export --list
```

```
UUID                               Name
```

```
-----
```

```
808a59a6d53b4f9ebfa3cb8df75ef8f3  cc-setup-vm
```

```
$ mkdir cc-setup-vm
```

```
$ sudo /home/hofmannp/go/bin/ovf-export -id 808a59a6d53b4f9ebfa3cb8df75ef8f3 -output ~/cc-setup-vm -ova
```

The `ovf-export` is available as open source [OVFEXPORT]. The following changes were necessary to make the software run on our system:

```
diff -ruw ovf-export.orig/qemu-utils/qemu-img-convert.go ovf-export/qemu-utils/qemu-img-convert.go
```

```
--- ovf-export.orig/qemu-utils/qemu-img-convert.go    2023-08-25 06:40:51.604966171 -0400
```

```
+++ ovf-export/qemu-utils/qemu-img-convert.go    2023-08-25 06:41:21.531023780 -0400
```

```
@@ -3,7 +3,7 @@
```

```
import (
```

```

    "fmt"
    "os/exec"
-   "runtime"
+//  "runtime"
    "strconv"
    "strings"
)
@@ -44,7 +44,7 @@
    vmdkopts = append(vmdkopts, "zeroed_grain")
}

-   var args = []string{"convert", "-m", strconv.Itoa(runtime.NumCPU()), "-O",
"vmdk"}
+   var args = []string{"convert", "-m", "16", "-O", "vmdk"}

    if len(vmdkopts) > 0 {
        args = append(args, "-o", strings.Join(vmdkopts, ","))

```

The software must be recompiled with

```
go install -a ./cmd/ovf-export
```

A.7 SETUP ON THE HOST SYSTEM FOR NEW GUEST VMS

The host admin (i.e. the infrastructure or cloud provider) must perform the following steps for setting up a new guest VM after having been notified by the guest owner that a new VM has been uploaded:

1. Copy any existing .xml file from an existing example (for Centos or Debian)
2. In the .xml file, rename the domain (global replace of name, should be 3 times)
3. In the .xml file, change the UUID so that it is unique
4. Adjust the path to the .qcow2 file
5. Copy the `/var/lib/libvirt/qemu/nvram/old_template_VARS.fd` file to `/var/lib/libvirt/qemu/nvram/new-name_VARS.fd`.
6. Change the QMP port to some unique value (will be used later for measured launch).
7. Open new QMP port on host firewall: `firewall-cmd --zone=public --add-port=XXXX/tcp --permanent; systemctl reload firewalld`
8. Make the domain known to libvirt: `virsh define new-name.xml`

After the Guest Owner provided the certificates for measured launch:

1. Paste the contents of the user-provided “session.b64” file into the `<session>` clause of the .xml file.
2. Paste the contents of the user-provided “godh.b64” file into the `<dhCert>` clause of the .xml file.
3. Test that the new domain can be started in paused mode: `virsh start --paused new-name`.

A.8 CONFIGURE THE REST SERVER AS SYSTEMD SERVICE

This activity needs to be carried out by the cloud or infrastructure provider to establish the remote management service for guest owners.

Create a file `/etc/systemd/system/remote-management.service` with the following contents:

```
[Unit]
Description=remote management daemon
After=network.target

[Service]
Type=notify
# the specific user that our service will run as
User=root
Group=root
WorkingDirectory=/usr/local/remote-management/rest_server
ExecStart=/usr/local/remote-management/env/bin/gunicorn -b
0.0.0.0:5000 app:app
ExecReload=/bin/kill -s HUP $MAINPID
KillMode=mixed
TimeoutStopSec=5
```

```
PrivateTmp=true
```

```
[Install]
```

```
WantedBy=multi-user.target
```

This requires the following:

- A chroot environment with the necessary python modules and the gunicorn server under `/usr/local/remote-management/env`.
- The prepared rest server installed under `/usr/local/remote-management/rest_server`.
- If there are problems with SE linux permissions issue:
 - `restorecon /usr/local/remote-management`
 - `chcon unconfined_u:object_r:bin_t:s0 /usr/local/remote-management/env/bin/*`

Enable and start the service like this:

```
systemctl enable remote-management
```

```
systemctl start remote-management
```

If any changes to the `user.py` configuration are made, restart the service:

```
systemctl restart remote-management
```

A.9 API DESCRIPTION

Guest owners need a remote access API to manage their VMs themselves. The interface features:

- Endpoints to execute the basic libvirt-based commands on a specified VM (start, shutoff, status...).
- Basic authentication for each endpoint: guest owner ID and password (assigned by DT), assigned VMs names.
- QMP protocol for measured launch.

The API is made using Python Flask framework for REST APIs (second most used Python framework for REST APIs and compatible with OpenAPI Generator). It needs to be run as root in order to execute libvirt commands.

The API structure is as follows:

rest_server/

├─ app.py (main file containing the API code)

├─ certificates.json (file containing certificates contents for each VM, required for measured launch)

├─ flask_api.yaml (file to generate API structure using OpenAPI Generator)

├─ users.py (file defining guest owner's usernames, password and assigned VMs names)

├─ ... files generated by OpenAPI Generator (not necessary)

Authentication

The API uses the HTTPBasicAuth library from Flask to authenticate users. Users' id, password and their assigned VMs names are defined in the "users.py" file. If the user doesn't have the right IDs or VM is not attributed to them when attempting to access to an endpoint, server will send an error 401 Unauthorized.

Generate API structure from YAML

Base structure of the REST API can be generated from the "flask_api.yaml" file using OpenAPI Generator. Endpoints and authentication are defined in the yaml file.

```
openapi-generator-cli generate -i home/cp/rest_api/flask_api.yaml -g python-flask -o rest_server
```

/!\ This only generate the API base structure, not the code for each endpoint

Launch API

```
cd /home/cp/rest-server/
```

```
flask run
```

!\ It is recommended to use a python virtual env to use Flask (`source /home/cp/env/bin/activate`) + and start the API as root.

Refer to appendix 0 for instructions to establish the service as a systemd unit.

Endpoints specifications

Every endpoint can be accessed with HTTP requests following this format:

- POST method
- "Authorization" header with user:password (can be done with the `--user` option of curl command)
- "Content-Type" header : 'application/json'
- VM name in request's body as json : `{"vm_name": "<vm_name>"}`

Template using `curl` command:

```
curl -X POST <host>/vm/status --user "<username>:<password>" -H 'Content-Type: application/json' -d '{"vm_name": <vm_name>}'
```

Available Endpoints:

- `/vm/start`: start VM in paused mode (uses `virsh start` command).
- `/vm/shutdown`: shutdown VM (uses `virsh shutdown` command).
- `/vm/destroy`: destroy VM (uses `virsh destroy` command).
- `/vm/status`: return VM status as `{"<vm_name>": "status" }` (uses `virsh dominfo` command).
- `/vm/certificates`: print given certificates content in the "certificate.json" file.

Obtaining and using the Docker Image for VM Operations

The Docker image is available from our download server as a TAR file "remote-management.tar". It can be imported into a local Docker or Podman installation like this:

```
docker load -input remote-management.tar
```

Guest owners can use the provided Docker image for remote management to manage their VM. The Docker container runs on CentOS 9 and has all the required software to run the

personalized commands (described below). Most commands require username and password of the guest to authenticate to API.

Necessary files (password file for API authentication, OVMF file...) for measured launch procedure are contained in the directory provided by DT. Please unpack the remote-management-workdir.tar in an appropriate location. The resulting workdir directory needs to be mounted to the Docker container as “/workdir” volume (using: `docker -v local_dir:/workdir ...`).

The following commands are available on the container (in the “/usr/local/bin” directory):

```
create-launch-bundle <launch-bundle-dir>
```

This command is used to create a launch bundle containing the required files for measured start of the encrypted VM. Requires the presence of the `pdh.cert` file provided by the remote host/cloud provider. It works as follow:

- Arguments:
 - `<launch-bundle-dir>`: the path to the directory where the files will be generated (directory will be created if not existing).
- Calls the `sevtool -generate-launch-blob` which “generate `launch_blob.bin`” and “`godh.cert`” certificates in the `launch-bundle-dir`. The files are then converted to “`session.b64`” and “`godh.b64`” files.

```
start <host> <vmname> <userid> <passwordfile>
```

This command is used to start the specified VM in paused mode (state required for measured launch). It works as follow:

- Arguments:
 - `<host>`: the address of the host.
 - `<vmname>`: the name of the VM to start in paused mode.
 - `<userid>`: the username used to connect to DT API.
 - `<passwordfile>`: the path to the passwordfile used to connect to DT API.
- Sends a call to API’s endpoint “/vm/start” with VM name in the request’s body.

```
launch <passwordfile> <create-launch-bundle directory> <qmp-socket>
```

This command is used to do a measured launch of the specified VM. It requires that the VM has been previously started in paused mode, and certificates were sent to the provider. It works as follow:

- Arguments:
 - `<passwordfile>`: A file containing the high-entropy password of the encrypted VM, as a String.
 - `<create-launch-bundle directory>`: the path to the launch-bundle-directory previously created.
 - `<qmp-socket>`: the qmp-socket port of the VM (given by DT).
- Does a measured launch of the VM through QMP protocol (does not call the REST API).

```
send <host> <vmname> <userid> <passwordfile> <create-launch-bundle-directory>
```

This command is used to send the content of the documents necessary to do a measured launch to the host (godh.b24 and session.b64 file). It works as follow:

- Arguments:
 - `<host>`: the address of the host.
 - `<vmname>`: the name of the VM to start in paused mode.
 - `<userid>`: the username used to connect to DT API.
 - `<passwordfile>`: the path to the passwordfile used to connect to DT API.
- Sends a call to API's endpoint `/vm/certificates` with VM name and files' content in the request's body.

```
destroy <host> <vmname> <userid> <passwordfile>
```

This command is used to destroy the specified VM (forceful shutdown). It works as follow:

- Arguments:
 - `<host>`: the address of the host.
 - `<vmname>`: the name of the VM to start in paused mode.
 - `<userid>`: the username used to connect to DT API.
 - `<passwordfile>`: the path to the passwordfile used to connect to DT API.
- Sends a call to API's endpoint `/vm/destroy` with VM name in the request's body .

```
shutdown <host> <vmname> <userid> <passwordfile>
```

This command is used to shut down the specified VM. It works as follow:

- Arguments:
 - `<host>`: the address of the host.
 - `<vmname>`: the name of the VM to start in paused mode.
 - `<userid>`: the username used to connect to DT API.
 - `<passwordfile>`: the path to the passwordfile used to connect to DT API.
- Sends a call to API's endpoint `/vm/destroy` with VM name in the request's body.

```
status <host> <vmname> <userid> <passwordfile>
```

This command is used to retrieve the current status of the specified VM. It works as follow:

- Arguments:
 - `<host>`: the address of the host.
 - `<vmname>`: the name of the VM to retrieve status from.
 - `<userid>`: the username used to connect to DT API.
 - `<passwordfile>`: the path to the password file used to connect to DT API.
- Sends a call to API's endpoint `/vm/status` with VM name in the request's body.

APPENDIX B: SUBJECTIVE TEST PLATFORM

B.1 PLATFORM INSTALLATION

B.1.1 System Requirements

This platform was developed on a system with an i9-13900K CPU, 64 GB of DDR5-4800 MHz memory and an NVIDIA RTX 4070 Ti. This is a high-end system (as of mid-2023), but the single most important requirement here is the memory size: at least 32 GB of main memory is recommended. Having a GPU with ≥ 8 GB of memory will also be beneficial.

B.1.2 Software Pre-requisites

The project has been tested with Unity version 2021.3.19f1. Please only use the same, as using newer versions can introduce bugs.

The project uses MRTK2 to work with the HoloLens 2. Check the tools needed to use MRTK 2 here⁹ and install them.

B.1.3 Preparing the Data

The point clouds PLY files need to be in binary little-endian format. This platform uses "Pcx - Point Cloud Importer/Renderer for Unity"¹⁰, and it only works with that format.

The meshes need to be generated offline. Ensure the meshes have the correct materials associated with them in Unity! The platform was developed and tested with OBJ files and works well with them.

One PLY or OBJ file per frame is utilised to animate the object on the screen.

B.1.4 Setting Up the Project

1. Download this repository and extract it.
2. Place the prepared point cloud files in the `Assets\Resources\Point-Clouds\<Name_of_point_cloud>\<Name_of_quality>\PointClouds`.
3. Similarly, place the prepared mesh files in the `Assets\Resources\Point-Clouds\<Name_of_point_cloud>\<Name_of_quality>\Mesh`.
4. Select to `PointClouds->Update Point Clouds From Assets` in the menu bar to let the project configure itself using the added objects.
5. MRTK should already be set up correctly but double-check the XR settings according to the MRTK documentation.
6. Enable/disable Holographic Remoting¹¹ as you desire.

⁹ <https://learn.microsoft.com/en-us/windows/mixed-reality/develop/install-the-tools?tabs=unity>. Accessed 02 November 2023.

¹⁰ <https://github.com/keijiro/Pcx>. Accessed 02 November 2023.

¹¹ <https://learn.microsoft.com/en-gb/windows/mixed-reality/mrtk-unity/mrtk2/features/tools/holographic-remoting?view=mrtkunity-2022-05>. Accessed 02 November 2023.

B.1.5 Point Clouds Loader

For both functionalities of this platform, there will be a `Manager` object in their respective scene. It contains a `Point Clouds Loader (Script)` component. It is responsible for loading the point clouds/meshes into memory.

Add new elements to the `Pc Objects` array in this component and provide information regarding the object name and quality levels present in the project. Only the objects and qualities mentioned here will be loaded into memory and are the only ones that can be used/interacted with.

The `Load Meshes` toggle controls whether the mesh files for the specified objects will be loaded.

Other values in this script should not be changed.

B.2 PLATFORM USAGE

B.2.1 Point Clouds Preview

The `Point Cloud Preview` scene can be found in `Assets_ConfigurationScene\ConfigurationScene.unity`.

1. Configure the `Point Clouds Loader` as desired.
2. The distance slider min and max values can be changed by updating the associated values in the `Distance Changer (Script)` component.
3. Run the scene.

The user can see and control the objects appearing on the screen. Up to 4 objects can be displayed and configured individually. The animation and interaction can be toggled.

B.2.2 Subjective Testing

The `Subjective Testing` scene can be found in `Assets_SubjectiveTesting\SubjectiveTestingScene.unity`.

Configure the `Point Clouds Loader` as desired.

B.2.2.1 Configuring the Test

The `SubjectiveTest` object in the scene has an `ST Manager (Script)` component. It is used to configure the tasks.

Open the `Tasks` array in the component and add task elements. Select the desired point clouds, representations, distances and qualities for each task. Refer to our paper [23] for more details.

The `Randomise Tasks` toggle controls whether the tasks are randomised. The sequences within the tasks are always randomised.

The `Use Fixed Y Offset For Distance` will add a height offset equal to `Y Offset` (in meters) to all objects displayed. You can use this to make your objects appear as if they are standing on the ground.

B.2.2.2 Running the Test



A `Start` button will be displayed before the start of each task. After every sequence, the test participant will be asked to give feedback between 1 and 10 using an immersive slider. The feedback is stored in the `Assets\CSV\ratings.csv` file.

