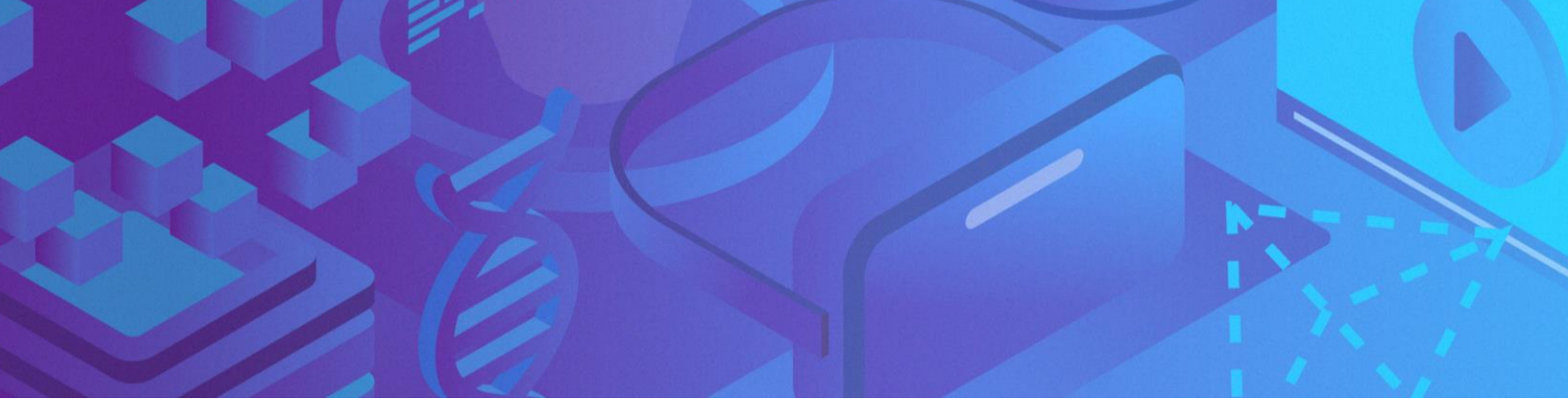

D2.2 USE CASE REQUIREMENTS, SYSTEM ARCHITECTURE AND INTERFACE DEFINITION (SECOND VERSION)

Work package	WP2
Task	Tasks 2.1, 2.2
Due date	30/09/2024
Submission date	30/09/2024
Deliverable lead	University of Surrey
Version	1.0
Authors	Ning Wang (UoS), Tim Wauters (IMEC), Sergio Tejeda Pastor (HHI), Nick Turay (Ericsson), Ali El Essaili (Ericsson), Christoph Stielow (DT), Peter Hoffmann (DT), Shiv Vats (UNI-KLU), Carl Udora (UoS), Taras Motulski (AWTG)
Reviewers	Hermann Hellwagner (UNI-KLU), Vivien Helmut (DT)
Abstract	This deliverable presents the updated version of the SPIRIT use cases by introducing an integrated application scenario of many-to-many telepresence together with the associated new requirements. An updated SPIRIT architecture and API definitions are also included in this document.
Keywords	Telepresence, use cases, network requirements, transport requirements, application requirements, security, architecture design.

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

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

DISCLAIMER

COPYRIGHT NOTICE

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	*Public, fully open, e.g. web (Deliverables flagged as public will be automatically published in CORDIS project's page)*	✓
SEN	*Sensitive, limited under the conditions of the Grant Agreement*	
Classified R-UE/ EU-R	*EU RESTRICTED under the Commission Decision No2015/ 444*	
Classified C-UE/ EU-C	*EU CONFIDENTIAL under the Commission Decision No2015/ 444*	
Classified S-UE/ EU-S	*EU SECRET under the Commission Decision No2015/ 444*	

* 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.

Funded by the Horizon Europe Framework Programme of the European Union

Funded by the Horizon Europe Framework Programme of the European Union

EXECUTIVE SUMMARY

This deliverable provides the updated version of the requirement analysis through a set of identified telepresence use cases of the SPIRIT project, which is followed by the introduction of the overall SPIRIT architecture. First, we provide the updated descriptions of the four project-defined use cases with the inclusion of additional requirements. The objective is to comprehensively identify and analyse the application and system requirements for a wide range of telepresence application scenarios. We also present the specification of the evolved SPIRIT architecture that provides a systematic view on different building blocks of the platform that will be used for supporting different application use cases. The explanation of a set of interfaces (APIs) is also provided in this deliverable.

Funded by the Horizon Europe Framework Programme of the European Union

TABLE OF CONTENTS

Funded by the Horizon Europe Framework Programme of the European Union

LIST OF FIGURES

Funded by the Horizon Europe Framework Programme of the European Union

ABBREVIATIONS

2D - Two-Dimensional

3D - Three-Dimensional

5G - Fifth Generation

6DoF - 6 Degrees of Freedom

AMR - Autonomous Mobile Robot

API - Application Programming Interface

AR - Augmented Reality

CUDA - Compute Unified Device Architecture

CPU - Central Processing Unit

DASH - Dynamic Adaptive Streaming over HTTP

DDOS - Distributed Denial of Service

DoS - Denial of Service

ECN - Explicit Congestion Notification

eMBB - Enhanced Mobile Broadband

FPS - Frames Per Second

GDPR - General Data Protection Regulation

GPU - Graphics Processing Unit

ICE - Interactive Connectivity Establishment

IDS - Intrusion Detection System

IMU - Inertial Measurement Unit

IP - Internet Protocol

KPIs - Key Performance Indicators

LiDAR - Light Detection and Ranging

LL-DASH - Low-Latency DASH

Funded by the Horizon Europe Framework Programme of the European Union

L4S - Low Latency, Low Loss, Scalable Throughput

MCU – Multipoint Control Unit

MEC - Mobile Edge Computing

MR - Mixed Reality

NFOV - Narrow Field-of-View

NR - New Radio

OS - Operating System

PC - Personal Computer

POI - Point of Interest

RAM - Random Access Memory

RGB - Red, Green, Blue

RTT - Round Trip Time

SAML - Security Assertion Markup Language

SCTP - Stream Control Transmission Protocol

SDK - Software Development Kit

SFC - Service Function Chaining

SFU - Selective Forwarding Unit

SSL - Secure Sockets Layer

STUN - Session Traversal Utilities for NAT

TCB - Trusted Computing Base

ToF - Time of Flight

TLS - Transport Layer Security

TURN - Traversal Using Relays around NAT

USB - Universal Serial Bus

VR - Virtual Reality

WSL - Windows Subsystem for Linux

WebRTC - Web Real-Time Communication

Funded by the Horizon Europe Framework Programme of the European Union

XR - Extended Reality

HTTPS - Hypertext Transfer Protocol Secure

Funded by the Horizon Europe Framework Programme of the European Union

1. INTRODUCTION

This document is composed as an updated deliverable from its first version (D2.1) in **Work Package 2 (WP2) "Requirements and platform architecture definition"** of the SPIRIT Project. It aims to elaborate the requirements associated with heterogeneous telepresence use cases and to introduce the overall SPIRIT architecture for facilitating the realisation of these use cases.

The use cases, detailed in Section 2 of this deliverable, are classified as follows:

(1) **Interactive human-to-human telepresence applications**. These use cases mainly focus on real-time streaming of telepresence media, but the specific scenarios range from immersive one-to-one interactions between remote end users, to the more complex case involving multi-party telepresence sessions across different geographical network locations. These use cases consider both avatar-based telepresence and direct streaming of captured human bodies, representing different application scenarios and requirements.

(2) **Real-time human-machine interaction applications**. This use case category considers the application scenario where end users rely on immersive telepresence applications to conduct live interactions with remote machines such as moving robots. In this case the key technical challenge is the necessity of performing real-time remote control over the machine in the field based on the immersive feedback from the live-streamed media.

In D2.1 we provided the descriptions of the basic features of the four use cases. In this document we introduce enhanced features on each of them with newly identified requirements (Section 3). Finally, in Section 4 we introduce the enhanced design of the overall SPIRIT architecture, which encompasses all necessary system components and interfaces required to realise the identified use cases in real-life environments. This generic architecture can be instantiated based on specific use case scenarios, thus helping the project team to implement and demonstrate them based on tailored support from the underlying system components. The architecture presented in this deliverable will be finalised in D2.3.

Funded by the Horizon Europe Framework Programme of the European Union

2. SPECIFICATION OF USE CASES

Four application use cases are being developed by the project team within the SPIRIT framework. In D2.1 we provided the description on the interim design of these use cases. In this deliverable we will expand them with distinct new features and requirements. These scenarios will be then complemented by those provided by the Open Call participants. There are two main categories regarding the use cases: real-time human-to-human communication and real-time human-to-machine interaction.

- **Real-time human-to-human interactions**

One of the use cases considered in the project is the real-time communication between two humans through the use of Extended Reality (XR) devices and the three-dimensional (3D) representation of one of the peers. Figure 1 shows a variety of scenarios to which the developments carried out in this project can be applied.

FIGURE 1: SKETCH OF POSSIBLE USE-CASES ENABLED BY REAL-TIME TELEPRESENCE COMMUNICATIONS

This type of real-time communication applications involves, in general, specific requirements such as low latency and high bandwidth. Several state-of-the-art technologies will be used to overcome these difficulties and guarantee a high-quality experience by the users. A combination of novel compression and rendering techniques together with low-latency streaming mechanisms will be key to achieve the necessary performance goals.

On another level, the realism in the representation of the human peers plays a fundamental role when creating an immersive telepresence environment. This challenge is addressed in different ways within the project, with a variety of scenarios that use the generation of 3D holographic representations of the human peers or realistic avatars animated in real-time from a media source.

In the use case descriptions in D2.1, we considered the simple scenario of one single viewer (receiver) in the corresponding telepresence applications. In this deliverable we extend the use case complexity by supporting multiple viewers at the same time. This newly added feature introduces additional technical challenges including the following: (1) capability of supporting one-to-multiple or multiple-to-multiple transmission of real-time telepresence content, (2) tailored content treatment (e.g. adaptations) according to specific network conditions on different viewers' sides, and (3) the increased complexity of resource management.

- **Real-time human-machine interactions**

The project also made efforts in exploring human-machine interactions. For this, a use case for intralogistics was devised to enable remote steering of Autonomous Mobile Robots (AMRs).

Funded by the Horizon Europe Framework Programme of the European Union

The use case describes a human-initiated scenario to teleoperate multiple robots in different locations providing flexibility in accessing and controlling robots in diverse environments, reducing the need for physical presence. The goal is to give the operator the necessary tools to safely control the robot from far away. A variety of different sensors like camera and Light Detection and Ranging (LiDAR) have been utilised to achieve this.

For this specific type of human-machine communication two major challenges have been identified.

The first challenge is to provide all the necessary information of the robot's surroundings to the operator in a time critical manner so that the operator can make useful decisions based on this information. For the teleoperation of robots this means that the data needs to be transmitted and displayed in real time. This problem becomes even more challenging when thinking about multiple sensors and even multiple robots transmitting data in real time.

The second major challenge identified is about the actual operation of the robot. How can the operator interact with robots in a safe way? Just like the presentation of sensor data, it is important that the drive commands coming from the operator are executed in near real time. This is important, because it again gives the operator a chance to make useful decisions based on the information he/she has in this very moment. Furthermore, it can be beneficial for the operator to have multiple helping mechanisms implemented to minimise the risk of accidents happening. One example is the emergency stop that stops the robot whenever something gets too close to the robot. Another good example would be an automated fine positioning which is not only safer but probably also faster than doing it manually.

Before introducing specific application use cases, we introduce in Table 1 the definitions of common key performance indicators (KPIs) associated with the use cases. When we present the description on each of the use cases in the rest of the section, we will also show the performance requirements (i.e. targeted values) against these KPIs where applicable.

KPI Label	Short	Detailed Description
RLat	End-to-end latency	Maximum allowed latency between input (audio, image, video, control) capturing and rendering on output devices
RDown	Download bandwidth	Minimum required data rate on the viewer side
RUp	Upload bandwidth	Minimum required uplink data rate on the producer side
RClients	Number of clients	Minimum supported total number of clients including sources and viewers
RSignal	WebRTC signalling server	Need for a WebRTC signalling server - maybe STUN/TURN - to establish a WebRTC connection successfully
RQoE	Target QoE value on a 5-point Likert scale	Value to specify the targeted Quality of Experience. The 5-point Likert scale offers two extreme, two intermediate, and one neutral response option to survey participants to express their attitudes accurately.
RFPS	Video frame rate	Minimum supported frame per second (FPS) rate at the viewer's device
RRes	Screen resolution	Minimum resolution level that should be supported on the viewer's device

Funded by the Horizon Europe Framework Programme of the European Union

RSync	Synchronisation	Maximum allowed time gap on the frame arrival across all the sources. Measured on the viewer's device side or the aggregation MEC server on the viewer side.
RInDev	Input devices	Variety of supported producer devices and their capabilities
ROutDev	Output devices	Variety of supported viewer devices and their capabilities
RFormats	Display formats	Ability to support coexistence of different display formats including point cloud and avatar based objects

TABLE 1. COMMON KPI DEFINITIONS FOR USE CASES

Table 2 below provides a high-level description of the four use cases, and in Section 2.1 and Section 2.2, detailed information on them will be provided.

UC-ID	Use Case Name	Short Description
Multi-Source	Live Multi-Source Holographic Streaming	Live teleportation with Fifth Generation (5G) MEC support: 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.
Avatar	Real-Time Animation and Streaming of Realistic Avatars	A scenario in which the avatar is animated by an animation library that makes use of a neural network. The input to this network is media captured on a mobile device. The rendering of the object is split between a cloud server and the consumer client device, to reduce the amount of data to be transmitted.
Holograms	Holographic Human-to-Human Communication	The primary focus of holographic communication lies in real-time 3D human interaction use cases that capture facial expressions and features during conversational calls, considering asymmetrical communication where one person is captured and displayed as a hologram to others. Extensions to this use case involve bi-directional conversations between multiple clients, as in many-to-many conferencing.
Robot	Distributed Steering of Autonomous Mobile Robots (AMRs)	An AMR transports goods from A to B. Using computer screens, the operator can control the AMR. The mounted cameras and sensors on the AMR allow the operator to have the full vision of the vehicle on their screen.

TABLE 2 SHORT DESCRIPTION OF USE CASES

2.1 REAL-TIME HUMAN-TO-HUMAN INTERACTION

In the digital age, real-time human-to-human interactions have become a cornerstone of our daily lives, fundamentally altering the way we communicate, connect, and collaborate. This paradigm shift is characterised by the immediacy and fluidity with which individuals can engage with one another, breaking down geographical barriers and fostering a sense of instant connectivity.

Funded by the Horizon Europe Framework Programme of the European Union

The rise of augmented reality (AR) and virtual reality (VR) technologies has taken real-time interactions to the next level. These immersive technologies offer a more natural and authentic experience by capturing non-verbal cues, gestures, and spatial awareness, enhancing the overall quality of communication.

While real-time interactions bring numerous benefits, it is essential to address challenges such as bandwidth limitations, technological accessibility, and the potential for information overload. As technology continues to advance, these challenges are being met with innovative solutions, ensuring that real-time human-to-human interactions remain a driving force in shaping the future of communication.

2.1.1 Live Multi-Source Holographic Streaming (Multi-Source)

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. dance) 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.

2.1.1.1 Specification of the use case operation

Figure 2 below shows the overall framework of the end-to-end network support of live teleportation applications based on the open source platform LiveScan3D.

At the content source side (e.g. the person to be captured and teleported), multiple sensor cameras (e.g. Kinect Azure DK cameras) are used to capture the object from different directions and each camera is connected to a local Personal Computer (PC) which is responsible for local processing of the raw content data, which 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 production server can optionally be located at a 5G MEC site. As shown in Figure 2, a common 5G MEC server can be used for providing remote production services to all the regional clients. In this case local frames from individual clients are streamed in real-time to the local 5G MEC server through 5G new radio (NR) uplinks.

On the receiver (content consumer) side, a dedicated 5G MEC server can be used to perform real-time synchronisation on the incoming frames from multiple sources at different locations. The purpose is to eliminate possible consumer-perceivable motion-misalignment effects caused by uncertainties of frame arrival time. Such uncertainties can be caused by a wide range of factors such as path distances and conditions from different sources to the consumer side, as well as the working load of remote clients. The frame synchronisation operation is typically based on a simple algorithm for frame pairing approximation based on the timestamp embedded in each incoming frame. A detailed specification of the synchronisation function can be found in [2]. Finally, the synchronised frames from the local MEC can be streamed in real-time to local consumer devices such as Microsoft HoloLens devices through 5G NR downlinks (forwarded to the Head-Mounted Displays (HMDs) by Wi-Fi via tethering). In this deliverable, we further consider the support of multiple simultaneous viewers in addition to multiple sources with one-way content delivery. One representative application scenario is distributed virtual

Funded by the Horizon Europe Framework Programme of the European Union

performance applications where performers from different physical network locations can be teleported in real time to the audience (viewers/ consumers) at different locations.

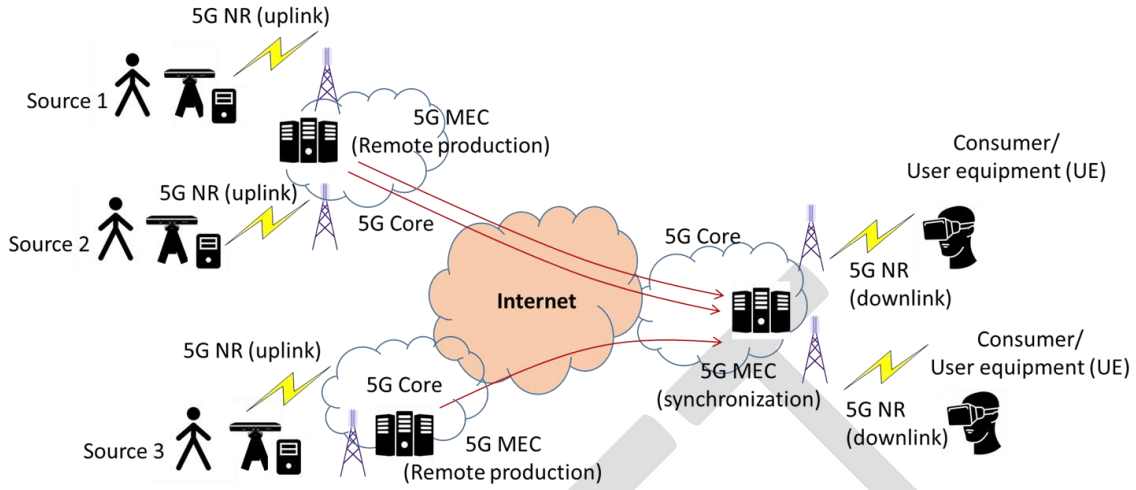

FIGURE 2: FRAMEWORK FOR MULTIVIEW UC

2.1.1.2 Application performance target of the use case

Table 3 shows the list of the KPIs and their targeted values for the multi-source use case. Below we further provide additional illustrations on a subset of them that require more explanation.

KPI Label	Targeted Value
RLat	200 ms
RUp	50 Mbps
RDown	50 Mbps Rate for minimum number of sources
RClients	4, including: two unique producers/sources and two unique viewers to demonstrate the scenario of multiple-to-multiple telepresence
RFPS	30
RRes	HD (1280x720 pixels)
RSync	50 ms
RQoE	4
RInDev	Kinect-2, Azure DK cameras
ROutDev	Hololens-2, Mobile phones, Proto_M Display

TABLE 3 KPI TARGETS FOR THE MULTI-SOURCE UC

- **RFPS:** This metric indicates the smoothness of the playback perceived by the consumer and typically it is represented as frames per second (FPS). The maximum FPS is normally determined by the camera capacity, e.g. the Azure DK cameras have the capacity of 30 FPS. It is also worth mentioning that network impairment may affect the FPS performance

Funded by the Horizon Europe Framework Programme of the European Union

that can be perceived on the receiver side as compared to the camera's FPS capacity. The general requirement is that the user received FPS should be statistically the same as the frame capacity at the camera side, meaning that the network delivery of these frames should not jeopardise the FPS performance from the source to the receiver side.

- **RLat:** This metric refers to the time gap between what is exactly happening in front of the cameras and the time point when the receiver actually sees it during playback. Same as conventional video applications, this is an essential metric for supporting interactive communications between end users. Specific to the scenario of multi-point to point teleportation without bi-directional interaction, there will be no stringent playback latency requirement. However, if the application involves multi-party interactions, then the playback latency will be essential which should be below 200ms. The playback latency perceived by the end user may consist of the latency contributions from the following: processing latency (encoding, decoding and frame buffering) and also the network latency between the source and the destination sides.
- **RSync:** This metric is specific to the multi-source teleportation scenario, which is about the synchronisation accuracy of playing back live frames streamed from different Internet locations. There has not been any subjective user QoE analysis on this, and it also depends on the movement of the objects being teleported. We believe that sub 50ms is reasonable for general frame synchronisation requirements.

2.1.1.3 Environmental requirements

- **Application hardware and configuration:** To support a 30 FPS frame rate at the FHD resolution level from multiple sources, the MEC server needs to have a minimum of two Nvidia A6000 GPU cards, complemented by a matching CPU and memory capacity. The MEC server should run the Windows Server 2019/2022 operating system and incorporate a virtual machine solution like the Windows Subsystem for Linux (WSL) to facilitate offline point cloud processing functions. On the content source side, for a single-user multi-view scenario, Kinect Azure DK cameras should be employed as capture devices. It is recommended to use no more than 4 cameras at FHD resolution, with each camera paired with a laptop that meets a minimum requirement of an Nvidia GTX 1060. On the user side, a HoloLens 2 or another HMD with equivalent point cloud data rendering capabilities is required.

- **Network platform and configuration:** The network should be a 5G network that adheres to at least Release 15 with a standalone architecture, allowing multi-user registration and data plane establishment. The access type should be 5G Enhanced Mobile Broadband (eMBB) as the radio access type to guarantee sufficient bandwidth for video streaming in both uplink and downlink directions. The split ratio of uplink and downlink bandwidth can be adjusted based on the user's role in sending or receiving holograms. The 5G terminal should be situated at the optimal line-of-sight according to the base station antenna angle, ensuring that no obstructions hinder the signal path. The HoloLens 2 headset needs to be tethered to a 5G phone for user session registration and initialisation in the 5G network. Assuming the captured content is an adult standing two meters away from the camera, with the ZSTD [3] compression algorithm enabled, different resolutions demand varying network resources (like bandwidth)..

- **Software functionality and configuration:** On the content source side, LiveScan3D [1] (referred to as the LiveScan3D client) should be deployed alongside the Microsoft Kinect SDK and Body Tracking library [4] to support real-time point cloud capture. The captured

Funded by the Horizon Europe Framework Programme of the European Union

content will be filtered by a pre-trained deep learning algorithm to detect and isolate the human body. The filtered point cloud data can either be saved to a local disk or directly transferred to a remote MEC server. The client will operate as a TCP client by default, proactively connecting to the remote MEC TCP server. The MEC server will deploy LiveScan3D [1] (referred to as the LiveScan3D server) to request/receive, process, calibrate, and merge content from one or multiple sources, or to transmit it to user devices. Additionally, the MEC server is equipped to display the real-time rendered point cloud data. On the user side, the HoloLens 2, equipped with TCP connectivity and a Unity/MRTK-based [5] point cloud rendering function, can receive and display the point cloud in real-time.

- **Performance enhancement features:** To offer enhanced user interactive functionalities with multisensory feedback, such as haptic feedback, the laptop on the content source side can be paired with additional hardware, like a haptic glove. To enhance performance robustness amid network uncertainties, features such as Multi-TCP connections and intent-aware network adaptation can be integrated into the platform as needed. Additionally, the platform supports a RESTful API, providing an interface that authenticated external entities can use to query and modify content and network settings, such as resolution levels.

- **Support of multiple concurrent viewers**: To minimise the network platform complexity, by default we rely on one centralised server for frame synchronisation, as shown in Figure 2. Specifically, each of the UE devices are connected to the common server from where the live frames originated from different remote network locations have already been synchronised. From the receivers' point of view, their network distance to the server might be different, however it is expected that this will not cause issues of frame synchronisation for each of them, as the frames will be transmitted through a common delivery path to the UE as the final destination. In the context of multiple viewers, the new KPIs will include the number of concurrent receivers (which is 2) and the fulfilment of their individual user-experiences KPIs as indicated previously.

2.1.2 Real-time Animation and Streaming of Realistic Avatars (Avatar)

The realistic volumetric representation of human beings has become an important topic in the last few years. The increasing demand of real-time telepresence scenarios presents a valuable opportunity to develop applications that portray the users in a much more immersive way than how it had been done in the past. This, in combination with Mixed Reality (MR) devices, guarantees a better overall experience.

Such applications, however, present novel technical challenges that need to be overcome to ensure a fluent and comfortable communication. The significant amount of data that a photorealistic avatar comprises, and the variable conditions of the network are two of the hurdles that need to be addressed. Besides this, the animation of the three-dimensional object must be done in real-time, taking some kind of media as an input (audio, video, text, 6 Degrees of Freedom (6DoF) position). This brings additional flexibility to the representation.

This use case proposes a scenario in which the avatar is animated by an animation library that makes use of a neural network. The input to this network is media captured on a mobile device. The rendering of the object is split between a cloud server and the consumer client device, to reduce the amount of data to be transmitted. In the network, low latency is guaranteed using congestion control algorithms. The local user device performs the integration of the avatar into

Funded by the Horizon Europe Framework Programme of the European Union

the real world and allows the user to interact with it. The following video shows an example of the application: Splitrendering_avatar_head_rotation

2.1.2.1 Specification of the use case

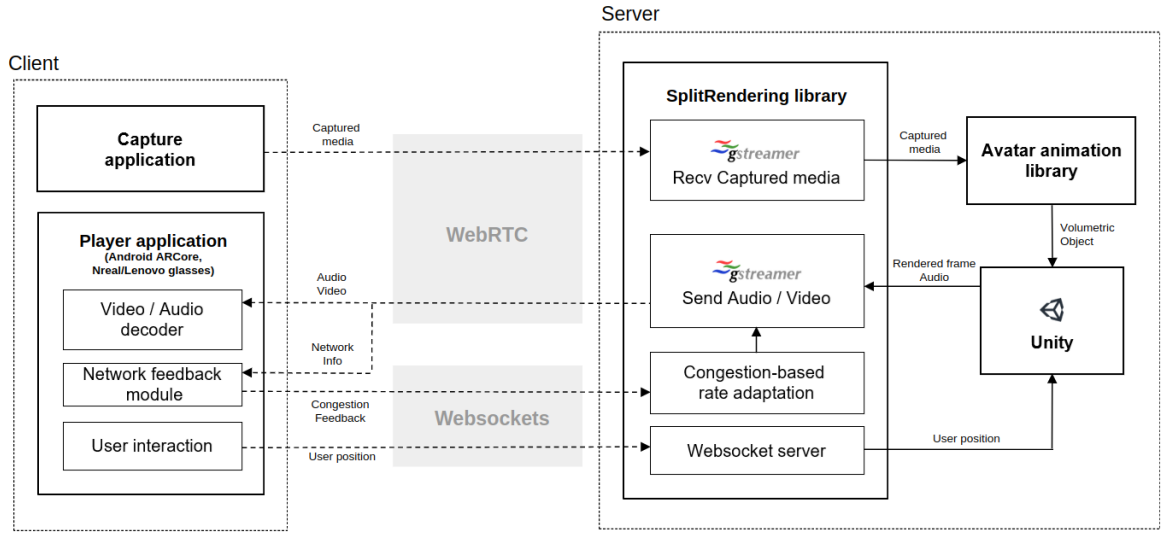

FIGURE 3: SYSTEM ARCHITECTURE OF AVATAR ANIMATION, SPLIT RENDERING AND STREAMING

The cloud-based streaming for avatar animation, illustrated in Figure 3, shifts the computationally intense rendering from the client to the MEC server, while providing a reliable interaction to users through low latency streaming. The *SplitRendering library* [6] on the server manages to transport media / user metadata via Websockets and WebRTC over the network. It receives the captured media (i.e., recorded audio) from the client and feeds it into the avatar animation library, a media-driven component which generates non-pre-captured real-time interactive animation. The rendering engine (Unity) renders the interactive avatar and generates a two-dimensional (2D) image of the 3D object from the proper angle, according to the current position and orientation of the final user. This 2D image is then compressed by a hardware encoder (e.g., NVEnc) and transmitted to the client.

The *Player application* on the client renders the 2D video synchronised with the user viewport on AR / VR mobile devices by leveraging existing 2D video decoding codecs such as H.264 or H.265. The original solid colour present in the background of the rendered image is here removed, achieving a better integration of the avatar in the real scene. At the same time, this client application sends information about the position and rotation of the avatar to the server, to synchronise the rendering view and provide the feeling that the whole 3D object is being rendered in the client device, although only 2D images are being received, therefore reducing the required bandwidth significantly. On this application, the user can as well interact with the avatar by scaling it and adjusting its position. The avatar addresses the user by adjusting the head rotation to ensure that there is a natural eye contact in the scene.

The network space is considered to be in a mobile network with the bottleneck being the access link between the mobile devices and the base station with Explicit Congestion Notification (ECN) marking to signal congestion severity to the client. The server adjusts the

Funded by the Horizon Europe Framework Programme of the European Union

encoding bitrate based on a rate adaptation algorithm (e.g. L4S - Low Latency, Low Loss and Scalable Throughput[1]) with the congestion feedback sent by the client.

As an extension to the basic use case, the possibility of connecting more than one consumer user is being addressed. In this scenario, each of the client applications will receive a different image of the avatar, depending on their current position in the scene. This will result in multiple video streaming channels that must be rendered, encoded and transmitted simultaneously. On top of this, the interactions between the users and the avatar (such as the modification of its position) must be synchronised at the server in such a way that every other user is aware of the action with a latency as low as possible. The integration of this one-to-many configuration in the existing Split Rendering application will be the main challenge, since the capabilities of the Nvidia hardware encoder to perform parallel encoding will be put to the test. Figure 4 shows this new architecture.

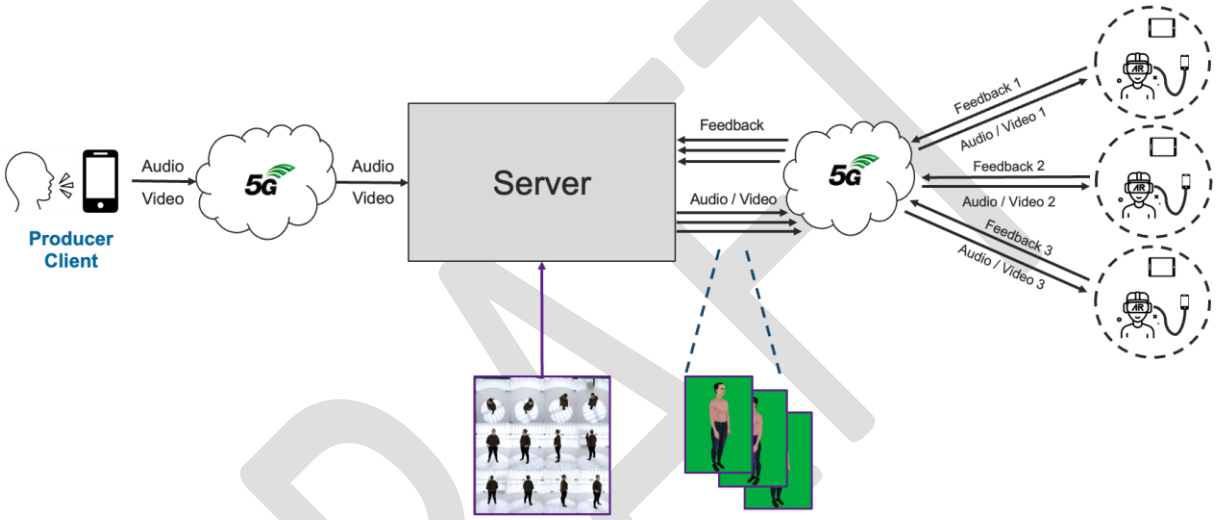

FIGURE 4 ARCHITECTURE OF THE ONE-TO-MANY SCENARIO FOR THE AVATAR USE CASE

2.1.2.2 Application performance target of the use case

KPI Label	Targeted value
RLat	200 ms The E2E latency must not be higher than this value in order to guarantee a fluent communication
RUp	> 5 Mbps for captured media
RDown	> 5 Mbps for Mono video (just one image played on a tablet/phone) > 10 Mbps for Stereo video (one image per eye played on XR glasses)
RClients	3 A producer client that sends audio and 2 receiver clients that play audio and video simultaneously
RFPS	25 The current animation for the avatar runs at 25 FPS, so the whole application uses this framerate
RRes	1920x1080 pixels

[1] https://datatracker.ietf.org/doc/rfc9330/

Funded by the Horizon Europe Framework Programme of the European Union

	This minimum resolution must be used to perceive a sufficiently photorealistic avatar
RInDev	Android Smartphone/Tablet compatible with WebRTC
ROutDev	Android 5G Smartphone/Tablet compatible with WebRTC, Unity and ARCore XR Glasses: XReal Light, Lenovo ThinkReality a3
RQoE	4

TABLE 4 KPI TARGETS FOR AVATAR UC

- **RRes:** The resolution of the 2D image received by the user can be selected prior to starting the server. The minimum functional resolution is 1920x1080, but higher (up to 4K) resolutions have also been tested successfully. When rendering the image on the server, the field of view of the camera is adapted to the distance from the user viewpoint to the avatar. The real resolution of the image that the user receives will then vary, to guarantee that the whole object is always shown.

- **RFPS:** The avatar animation library works at a frame rate of 25 FPS. Therefore, the provided streaming works at that rate as well.

- **RLat:** The motion-to-photon latency varies depending on the network conditions and the desired resolution. To guarantee the smoothness of the experience and the proper adaptation to the user's movement, the maximum acceptable latency is 200 ms.

2.1.2.3 Environment requirements of the use case

- **Application hardware and configuration:** The rendering functionalities of the use case require the server machine to be equipped with an Nvidia GPU card that can work with the Computed Unified Device Architecture (CUDA) platform (minimum version 11.3) and has at least 8 GB of Random Access Memory (RAM), with a Central Processing Unit (CPU) that can provide proper support to it. On the client side, the media capture device must be an Android phone or tablet with Android 7.0+. The player application must run on a device with Android version 8+ and capable of using ARCore. When using XR glasses, the list of compatible phones must be consulted for each individual model.

- **Network architecture and configuration:** As mentioned, the streaming pipeline of the system uses WebRTC to send audio, video, and data. The minimum required bandwidth is 10 Mbps or 15 Mbps if working with 4K resolution. Both wired and wireless connections can be used. For the latter, Wi-Fi or 5G are recommended. When streaming to a network outside the one that hosts the server, the ports used by WebRTC, Websockets, Interactive Connectivity Establishment (ICE) and Session Traversal Utilities for NAT (STUN) / Traversal Using Relays around NAT (TURN) servers must be open. In order to allow the use of congestion control mechanisms, such as L4S, the network must provide information about its congestion status though ECN marking.

- **Software functionality and configuration:** On the server side, Ubuntu 20.04 LTS (Long Term Support) must be available, as well as the corresponding Nvidia drivers and CUDA 11.3+. The streaming module of the server application uses Gstreamer to send and receive audio and video, so a version 1.20+ of this framework must be installed in the system. On the client, the tablet/phone player application requires ARCore to be installed on the device. For the XR glasses, the required software by the vendor needs to be set up prior to using the system.

Funded by the Horizon Europe Framework Programme of the European Union

- **Performance metrics:** The set of parameters that can be used to evaluate the quality of the streaming is based on the measurements of the received bitrate, link utilisation, latency (RTT – round trip time, one-way delay), packet loss and frame rate.

- **One-to-many capabilities:** new parameters will be required to evaluate the performance of the use case when more than one consumer client is connected to the server. These should include the number of clients supported by the application and the end-to-end latency for each of them.

2.1.3 Holographic Human-to-human Communication (Holograms)

Immersive telepresence, specifically holographic communications, denotes a technology enabling individuals to engage in three-dimensional communication using data formats such as point clouds or meshes. Unlike traditional communication methods like video or telephone conversations, holographic communication strives to provide a more immersive and genuine experience by streaming real-time 3D representations of individuals.

Following years of widespread adoption of video calls on smartphones and tablets, users express keen anticipation for digital interactions facilitated by immersive communication services, such as three-dimensional holographic AR calls over 5G networks [7]. Research [8]indicates that over 50% of smartphone users expect holographic communication technology to become accessible in the near future. With the expected rise in remote work in the coming decade, many professionals will seek more immersive digital interaction methods. A recent study identified the need for social interaction as a significant barrier to remote work [9]. This suggests a promising future for holographic communication across diverse domains. Anticipated applications span consumer and enterprise sectors, encompassing activities like participating in family gatherings through holographic representations, receiving medical consultations from home, engaging in remote office environments, accessing expert guidance in industrial settings, and immersive marketing experiences.

The primary focus of holographic communication lies in real-time 3D human interaction use cases that capture facial expressions and features during conversational calls, considering asymmetrical communication where one person is captured and displayed as a hologram to others.

2.1.3.1 Specification of the use case

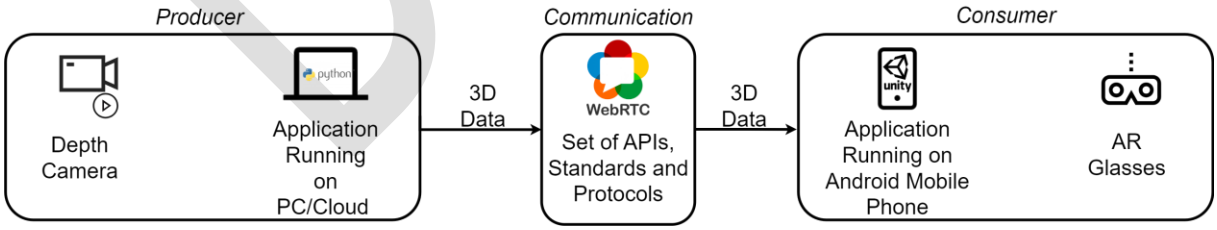

FIGURE 5: HOLOGRAPHIC COMMUNICATION USE-CASE OVERVIEW

Figure 5 shows the overall concept of the holographic communication use-case implementation.

The use case involves two users, with one user running a producer application on a PC or cloud system connected to a commercially available depth camera and one user running a consumer application on an Android mobile phone. The camera is equipped with a software

Funded by the Horizon Europe Framework Programme of the European Union

development kit (SDK) that provides application programming interfaces (APIs) for 3D data processing that can be used by the producer application based on the Python programming language. The consumer application is based on the Unity game engine offering various extensions for 3D processing as well.

In the use case, the face or upper body of the user on the producer side is captured by the depth camera, providing both colour and spatial information for the producer application, which is processed on the PC or cloud system running the application. The first processing step involves creating a point cloud using the synchronised RGB and depth images previously captured by the depth camera. The point cloud is then converted into a mesh, which is displayed in the form of surfaces. To do this, the edges between the vertex coordinates are calculated to form triangles from neighbouring points in the point cloud. The mesh containing geographical and texture information is then encoded in order to reduce the bandwidth required for data transmission while largely preserving the original quality of the mesh.

Once the mesh, hereafter referred to as 3D data, is compressed, the producer application starts streaming the 3D data to the user running the consumer application. The 3D data is streamed in a peer-to-peer process using Web Real-Time Communication (WebRTC) in conjunction with the TCP-based WebSocket protocol for signalling purposes. Signalling is carried out with the help of a signal server provided by the use case provider, namely Ericsson.

The user on the consumer side, who is equipped with an Android mobile phone connected to the AR glasses via USB-C, receives the 3D data stream. The first step involves decoding the data on the mobile phone. The 3D data is then filtered and rendered. The user on the consumer side then sees an authentic human 3D representation of the user on the producer side in real time via the AR glasses.

To further improve the QoE of the users within the use case, the producer application offers various options for adapting the 3D content to the available resources, e.g. filtering the RGB and depth frames to adjust the resolution of the final 3D representation, changing the intensity of the compression for data transmission or setting the spatial distance threshold between the object and the depth camera to remove the background pixel not of interest.

WebRTC is primarily a peer-to-peer communication protocol, allowing direct media exchange between browsers and devices without requiring central control. However, two key server-based architectures can be considered to improve scalability: selective forwarding unit (SFU) and multipoint control unit (MCU). The former receives media streams from all participants and forwards them selectively to others without altering the streams. This reduces bandwidth usage for each participant, but still keeps individual streams. The latter combines the media streams from all participants into a single stream before sending it back to the participants. This simplifies the processing on each participant's device but requires more server resources. As an extension towards bidirectional one-to-many and many-to-many capabilities, as in classroom and tele-conferencing scenarios, respectively, integration with the SFU and/or MCU, developed in WP3 for scalability, is envisaged.

2.1.3.2 Application performance target of the use case

KPI Label	Targeted Value
RLat	200 ms
RDown	20 Mbps Compressed meshes
RClients	2 (extension towards 10+) One producer and two receivers

Funded by the Horizon Europe Framework Programme of the European Union

RFPS	30 Typical user experience
RRes	1280x720 (Strongly dependents on the hardware the producer application runs on)
RQoE	4
RInDev	Intel real-sense depth camera
ROutDev	Android 5G smartphones / tablets AR glasses
RCPU	8-16 Producer side

TABLE 5 KPI TARGETS FOR HOLOGRAM UC

- **RLat:** In this use case of immersive human-to-human interaction, the playback latency, hereafter referred to as glass-to-glass delay, must have a maximum value of about 200 ms. This measurement encompasses source delay (camera data acquisition), processing delay (application software), and destination delay (frame rendering and visualisation), encompassing all latency paths except for network latency. Tests accounting for network latency are currently underway.

- **RDown:** For compressed meshes, e.g., using Draco encoding, a downlink bandwidth of at least 20 Mbps is required to support higher resolutions.

- **RFPS:** The maximum frame rate given in FPS is typically governed by the capabilities of the camera; for instance, the Intel RealSense camera can handle up to 90 FPS. In the case of the human-to-human interaction use case a value of 30 FPS was found to be satisfactory. However, it is crucial to minimise network and software disruptions to ensure a consistent FPS between the sending and receiving ends.

- **RRes:** The user has the flexibility to adjust the resolution in two ways. Firstly, assuming an Intel RealSense depth camera, directly through the camera API, offering a range of available resolutions from 320x180 to 1280x720 pixels.

  Alternatively, the second option involves employing decimation filtering in a software post-processing step on the device connected to the camera. Presently, the decimation filter algorithm operates by reducing the number of pixels per frame by half, and it can be easily customised to different filter ratios (reduction of original resolution) according to the specific requirements of the application.

2.1.3.3 Environment requirements of the use case

- **Application hardware and configuration:** On the producer side, a depth camera is utilised for gathering spatial and colour information about an object of interest. We highly recommend the use of an Intel RealSense depth camera, given that the application software was specifically developed and tested using the Intel RealSense SDK. The processing component of the producer side necessitates a PC or cloud system with a minimum of 32 GB of RAM, along with a CPU capable of providing adequate support to handle the CPU-intensive encoding phase of processing. It is crucial to emphasise that the Intel RealSense camera only supports a USB 3.0 port. Therefore, a system must have functional USB 3.0 ports. Alternatively, the camera can be connected to an Ethernet port via Universal Serial Bus to Internet Protocol (USB-to-IP) conversion using software or hardware solutions. On the consumer side, an Android mobile phone is essential for the

Funded by the Horizon Europe Framework Programme of the European Union

functionality of the application. During the development and testing phase, the application used XReal Light AR glasses to display 3D content from the mobile phone. We strongly recommend the use of the XReal Light AR glasses, as this ensures compatibility with the consumer application. It is important to note that compatibility of the application is limited to specific mobile phones when using XReal glasses. Therefore, it is imperative to consult the list of compatible phones to ensure seamless functionality.

- **Network architecture and configuration:** The network architecture must enable access to the Internet to establish a peer-to-peer connection using WebRTC founded by a signalling procedure using a signalling server hosted in an Ericsson network. Both wired and wireless network access technologies are viable options. For wireless links, Wi-Fi or 5G NR technology is recommended.

- **Software functionality and configuration:** The containerised software on the producer end, developed in Python, has been successfully implemented and tested on Windows 10 and Debian 13 operating systems. The container includes all necessary dependencies. On the consumer end, the mobile phone must operate on Android version 11 to ensure optimal performance. If needed the software can also be adapted to support more recent Android versions. Tests are ongoing. Additionally, for the AR glasses to function properly, it is essential to set up the required software provided by the vendor before using the system with a mobile phone.

- **Performance metrics:** The evaluation of the holographic streaming service relies on a set of parameters, including measurements of sent/received bitrates. Additionally, the evaluation considers individual latency components such as source jitter, application delay, and glass-to-glass delay. Work is currently underway to implement Key Performance Indicator (KPI) measurement tools to enhance the evaluation process further.

2.2 REAL-TIME HUMAN-MACHINE INTERACTIONS

2.2.1 Distributed Steering of Autonomous Mobile Robots (AMR) (Robot)

The demand for autonomous applications is on the rise to address increasing challenges like growing mobility pains in cities, skilled labour shortage, safer mobility, and enhanced logistics. An AMR transports goods from A to B. Complex scenarios like loading and unloading pallets can be a challenging task to fulfil, as the loads can vary in size and weight. For that reason, distributed steering is available for the operator. Using computer screens, the operator can control the AMR. The mounted cameras and sensors on the AMR allow the operator to have the full vision of the vehicle on their screen. After being temporarily controlled by the operator, the AMR can autonomously manage its route as usual. The operators can teleport between different vehicles, across multiple sites. Remote operations enhance workers' safety and increase productivity.

2.2.1.1 Specification of the use case

To drive around an area autonomously, the AMR needs to have a map. The map is created by the operator who is navigating the AMR remotely through the area. After the map is created, the AMR knows exactly within which area it can move around. No-go areas can be created within the map management. The AMRs are running in a typical production environment. The rectangular hall measures 30x40 meters (1200 square meters). An outside area is also

Funded by the Horizon Europe Framework Programme of the European Union

available. Points of interest (POI) are created within the map to send the AMR from one point of the hall to another. Using the mounted cameras and sensors, the AMR can detect obstacles, avoid them and replan its path accordingly. The AMR drives around autonomously but can also be steered remotely. Remote control is used in the case of loading and unloading or on special occasions that require a real person to intervene. The mounted cameras and sensors on the AMR give the operator full visibility to perform. One platform manages all AMRs from various manufacturers. Several handheld devices (computers, tablets, smartphones) are used for steering the AMRs, that are all operating in the same network. The objective is to demonstrate the ability to centralise steering of devices, in particular AMRs, into an edge computing device, while performing the steering over a wireless network. The data processing takes place in a local data centre (campus edge cloud) with a centralised device intelligence. The 5G communication works purely over a private network, through a local core network on campus, which is exclusively used for internal communication. There is no connection to the public mobile network / roaming.

2.2.1.2 Application performance target of the use case

KPI Label	Targeted Value
RLat	200 ms The E2E latency must not be higher than this value in order to guarantee a fluent communication
RDown	500 kbps for 1 camera 2 Mbps for 4 cameras
RUp	500 kbps for 1 camera 2 Mbps for 4 cameras
RFPS	30 The framerate is limited by the camera being used for the use case
RRes	The resolution of the received video streams is adjustable and supports up to 1280x720 pixels, Currently 640x480 pixels are used for the cameras.
RQoE	4
RInDev	Cameras supporting video4linux, Intel Realsense 435 is currently used
ROutDev	Any common device running a web browser

TABLE 6 KPI TARGETS FOR ROBOT UC

The operator sits in a room close to the motion area of the robots, using a multi-screen workplace to operate them when required. In areas with high demand for motion precision such as semi-structured warehouses or interaction with other road users, the robot needs to have sufficient upstream capabilities for video and sensor streams as well as sufficient downlink capabilities on the consumer site to be able to safely control and follow the robot's motion at all times.

- **RUp:**
  - Following an AMR's motion at all times in near real time per assisted camera-based telematics should be possible without interruption for safe operation.
  - For the operation of multiple cameras on multiple robots with regards to performance, the overall required upstream bandwidth should be kept small.
- **RLat:** This needs to be consistently below 200 ms to ensure safe operations with the robot.

Funded by the Horizon Europe Framework Programme of the European Union

- **RFPS:** The refresh rate of the use case is limited by the camera being used. In our case the RGB camera of the Intel RealSense 435 caps out at 30fps.
- **RRes:** The resolution of the received video streams is adjustable and supports up to 1280x720 pixels. The resolution is also dependent on the network and processing power.

2.2.1.3 Environment requirements of the use case

5G private campus network: A controller is used, which serves as the base unit for the 5G core baseband. It is the central unit that bridges the 5G network to the enterprise network. Radio equipment provides the signal for the dot antennas, which are placed within the environment. With the network management portal, the configuration and monitoring of the 5G site can be done remotely, to observe all devices that are connected. The services in the test network include IP-based packet-switched data traffic in the 5G network. They are spatially limited to the defined campus. When using data, the users booked in share the available bandwidth (so-called shared medium) in the campus mobile cells. Under ideal conditions, the campus network offers a network latency (RTT) of less than 20 ms, a downlink throughput of up to 1 Gbps and an uplink throughput of 100 Mbps. The achievable transmission speed during data usage depends, among other things, on:

- the frequency range used;
- the occupancy/utilisation of the mobile communication cell (the number of users);
- the distance to the antenna and the movement of the user;
- the terminal device used (including its operating system and other software used);
- the transmission speed of the dialled servers.

Edge computing: A local server that is on the campus providing scalable compute/storage options. The campus ecosystem with its edge computing platform can be seen in Figure 6. The edge computing platform is an open-source platform, where virtualisation and containerisation are provided. Central and decentral management is possible. Scaling edge-based navigation in real-time and knowledge sharing via remote edge cloud infrastructure is a key capability to provide seamless intralogistics use cases as well as real-time visibility of process deviations in intralogistics.

Control centre: The teleoperation use case runs in a Web-App which is why the control centre needs to be equipped with a device that is capable of running a Web browser, e.g. a phone, a tablet or a computer. Ideally, the control centre is also equipped with multiple computer screens so that the operator can observe multiple camera streams at once. The steering of the robot can be performed by either phone, tablet, computer keyboard or a controller that is connected to the computer.

Robot: A driving platform is needed to bring the use case to life. In order to be navigated, the robot needs to be connected and onboarded to the fleet management system. For the localisation and autonomous operation of the robot a map of the shopfloor needs to be recorded and uploaded to the server.

Cameras: To enable the teleoperation of the robot, cameras are needed on the robot to let the operator see the robot's surroundings. At the moment, 4 Intel RealSense 435 cameras are being used as RGB cameras; however, any camera supporting v4linux can be used with our teleoperation use case.

Funded by the Horizon Europe Framework Programme of the European Union

LiDAR: In order for the robot to record a map and localise itself in the map, it needs to perceive its environment. For this a LiDAR was mounted and integrated onto the robot. So far, we have used 2D LiDAR from SICK company and 3D LiDAR from Ouster company; however, theoretically any LiDAR could be integrated and used.

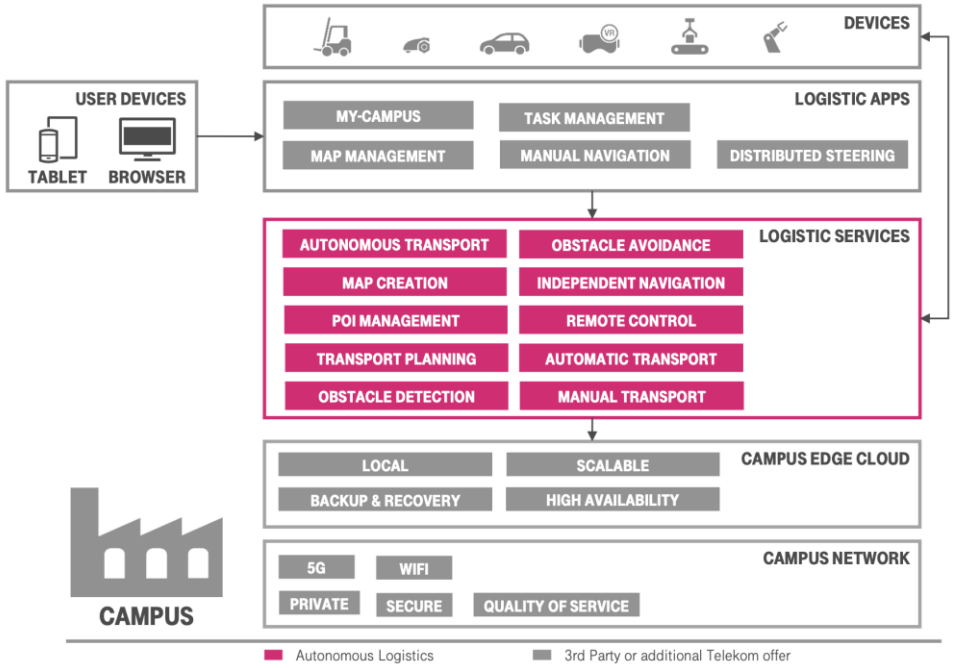

FIGURE 6: CAMPUS ECOSYSTEM

2.3 SUMMARY OF USE CASES

Label	Multi-source	Robot	Avatar	Holograms
RLat	< 200 ms	< 200 ms	< 200 ms	<200 ms
RDown	50 Mbps	> 2 Mbps	> 5 Mbps (Mono) > 10 Mbps (Stereo)	> 20 Mbps
RUp	50 Mbps	> 2 Mbps	> 5 Mbps	One-to-one unicast, so no RUp (extension towards bidirectional communication > 20 Mbps)
RClients	4 (extension towards multiple clients)	2	3	2 (extension towards 10+)
RSignal	no (for now)	no	yes	yes
RQoE	≥4	≥4	≥4	≥4
RFPS **RRes**	30 FPS	30 FPS ≤ 1280x720	25 FPS < 3840x2160	30 FPS 1280x720
RSync	<50 ms	-	-	-

Funded by the Horizon Europe Framework Programme of the European Union

RInDev / ROutDev	Input: Kinect-2, Azure DK cameras Output Hololens-2, Mobile phones, Proto_M Display	Input: cameras supporting video4linux Output: Any common device running a web browser	Input: Android Smartphone/Tablet compatible with WebRTC Android 5G Output: Smartphone/Tablet compatible with WebRTC, Unity and ARCore XR Glasses: XReal Light, Lenovo ThinkReality a3	Input: Intel real-sense camera Output: Android 5G smartphones / tablets XR glasses

2.4 POTENTIAL APPLICATION AREAS

In this section we elaborate on a variety of potential telepresence application areas beyond the project-defined use cases above. The purpose is to outline different categories of telepresence applications with different requirements that can be potentially supported by the SPIRIT platform. This also reflects the targeted use cases in different vertical sectors that the project would like to attract from third-party Open Call participants.

XR in Virtual Meetings

Create immersive and realistic virtual meeting spaces, allowing remote participants in different locations to feel as if they were actually in the same room.

- Provide a two-party or, further, a multi-party audio/video conference in high quality and a truly immersive user experience for the participants.
- Provide digital whiteboards that allow participants to collaborate on projects in real-time in brainstorming sessions and problem-solving.
- Enable 3D product demonstrations that allow participants to interact with products in a virtual environment. Target: sales teams or product designers who need to showcase their products to remote clients.
- Enable collaborative design on objects in real-time, even if the participants are located in different parts of the world, e.g., architects constructing buildings.
- Use AR to overlay information onto a physical object to create a more engaging and immersive presentation environment or to simulate maintenance procedures on complex machinery.

XR in Assistance and Collaboration

Facilitate real-time collaboration between technicians and support staff, allowing them to work together to diagnose and resolve issues in real-time. This can be especially useful for industries that require quick response times, such as healthcare or emergency services.

Funded by the Horizon Europe Framework Programme of the European Union

- Provide remote assistance to people in need. For example, a technician could use AR to guide someone through the process of fixing a piece of machinery.
- Provide virtual customer service experiences, allowing customers to interact with support staff in a more immersive and engaging way.
- Create immersive product design environments, where teams can work together to sketch out ideas and concepts in a virtual space.
- Facilitate real-time collaboration between technicians and support staff, allowing them to work together to diagnose and resolve issues in real-time to ensure quick response times, e.g., in healthcare or emergency services.
- Provide remote maintenance services, allowing technicians to perform routine maintenance tasks or repairs from a remote location.

XR in Training and Education

Create realistic simulations that allow trainees to practice real-world scenarios in a safe, controlled environment. This can be especially useful for industries that require hands-on training, such as healthcare or manufacturing.

- Provide remote training and education services, allowing technicians and support staff to learn new skills or procedures from a remote location.
- Create immersive learning experiences that engage trainees in a more interactive and memorable way. For example, explore different parts of the human body or learn how to operate complex machinery.
- Create historical reenactments that allow users to experience different periods in time. For example, pupils can explore ancient ruins or walk through historical landmarks.
- Create cultural immersion experiences that allow users to learn about different cultures and customs. For example, explore different types of food or learn about traditional clothing.
- Create virtual role-playing scenarios that allow trainees to practice real-world skills in a simulated environment, e.g., customer service or sales.

Funded by the Horizon Europe Framework Programme of the European Union

3. REQUIREMENTS AND RECOMMENDATIONS

Based on the identified use cases in Section 2, we provide the summary of general requirements and recommendations derived from the use cases. By *requirements* we mean necessary prerequisites that need to be fulfilled by the project team in order to fulfil the implementation and demonstration of the use cases. For more advanced features, especially those associated with final commercialisation of the products, we refer to them as recommendations, where applicable, that may be left open for future investigations even beyond the SPIRIT project lifetime. The organisation of such requirements and recommendations is based on the differentiation of the following technical areas: Network, Transport, Application, Security, and General.

3.1 NETWORK REQUIREMENTS

Network requirements

- Telepresence applications require a network with **low latency** to ensure that applications can respond quickly to user input. An end-to-end latency of less than 50 ms is considered ideal for telepresence applications.
- The SPIRIT platform should have a maximum end-to-end application latency of 100 ms for real-time interaction. This includes the time it takes to send and receive data to and from the server and display the results to the user. Low latency is critical for a smooth and responsive telepresence experience.
- Telepresence applications require a network with **high bandwidth** to transfer large amounts of data such as 3D models, textures, and videos in real-time. Regarding the bandwidth to support a smooth telepresence experience, it is largely depends on the specific application platforms for which the bandwidth requirements can range across the order of tens of Mbps.
- The SPIRIT platform should have a minimum bandwidth requirement of 10 Mbps for basic functionality. This includes streaming of telepresence content, downloading assets, and sending and receiving data to and from the server.
- Telepresence applications require a **reliable network connection** to ensure that the application does not get disconnected or interrupted during use. A network connection with low packet loss and high stability is important for telepresence applications.
- Telepresence applications should work on both **cellular data and Wi-Fi** networks. Cellular data networks are suitable for outdoor use, while Wi-Fi networks are often used indoors.

3.2 TRANSPORT REQUIREMENTS AND RECOMMENDATIONS

Transport requirements

- Telepresence applications should support multiple users, beyond mere one-on-one scenarios. One-to-many (e.g. concerts, events, classrooms) or many-to-many (e.g. large-scale teleconferencing) scenarios should be enabled. The WebRTC protocol

Funded by the Horizon Europe Framework Programme of the European Union

suite typically focuses on peer-to-peer connections, but more scalable solutions can be built using Selective Forwarding Units (SFU), media server components that can be deployed for limiting bandwidth usage and client connections for adaptive video streaming. Adapting such components for immersive media formats is an open research question.

Transport recommendations

- Enable the SPIRIT platform to cache data locally to reduce the amount of data transmitted over the network. This includes caching of content, assets, and user data. Caching should be optimised to reduce the amount of data transferred.
- Consider more scalable transport solutions, e.g. Low-Latency MPEG-DASH (Dynamic Adaptive Streaming over HTTP), not based on peer-to-peer connections (as is WebRTC). The key issue to investigate here is whether sufficiently low end-to-end latencies can be achieved for smooth telepresence applications.

3.3 APPLICATION REQUIREMENTS

- In practice, telepresence applications should support the use case scenarios **involving multiple simultaneous receivers**. There are two distinct cases that can be envisaged: (1) sources and receivers are distinct subsets with one-way transmission of telepresence content from the source side to the receivers' side (e.g. telepresence of virtual live performances); (2) telepresence participants are both sources and receivers in interactive virtual conferences.
- Be **compatible with a wide range of devices**, including smartphones and tablets with different specifications and operating systems.
- **Initialise and load** all required assets and modules in a reasonable time.
- Respond quickly to user input, such as touch, gesture, or voice commands. The **user input response time** should be less than 100 ms to ensure a smooth and responsive user experience.
- Enable telepresence applications to send and receive data from the server in real-time with minimal delay. The **network response time** should be less than 200 ms to ensure that applications operate smoothly.
- Enable telepresence applications to render virtual objects in real-time with minimal delay. The **object rendering response time** should be less than 50 ms to ensure that virtual objects appear seamlessly in the user's view.
- Enable telepresence applications to be **stable and reliable**, and they should not crash or freeze during use. The applications should be able to handle unexpected inputs and errors without interrupting the user's experience.
- Respond quickly to errors and provide appropriate feedback to the user. The **error response time** should be less than 500 ms to ensure that the user can continue using the application without interruption.

3.4 SECURITY REQUIREMENTS

Funded by the Horizon Europe Framework Programme of the European Union

The conditions to test, evaluate and validate the security requirements presented in this section are developed in section 7.

3.4.1 Requirements derived from the architecture

To support the SPIRIT security architecture as described in Section 4.2, the following concrete requirements have been identified for the SPIRIT project:

- **Strong encryption** of all communication paths in order to protect data-in-transit.
- Use of **confidential computing** mechanisms to protect data-in-use (specifically in the public cloud or exposed edge computing facilities).
- Implementation of an **intrusion detection system** (IDS) to protect the telepresence system.
- **Secure identification** of participants using identity management solutions tailored to the specific properties of telepresence systems.

3.4.2 Data protection and privacy

- **Inform users** about the types of data collected, such as location data, camera feed, or user input, and how the data will be used.
- **Obtain user consent** before collecting any personal data or tracking user behaviour. The consent should be explicit, informed, and revocable at any time.
- **Store user data securely** and protect it from unauthorised access, modification, or disclosure. The data should be encrypted at rest.
- **Ensure the integrity of user data**, including tracking data, user inputs, and server communications. The application should use encryption and other security measures to protect user data from unauthorised access or tampering.
- **Comply with data protection regulations,** such as the **General Data Protection Regulation (GDPR)**, and other regulations, depending on the type of data being collected.
- **Allow users to delete their data** and provide clear instructions on how to do so. The application should also ensure that the deleted data is completely removed from all backups and systems.

3.4.3. Secure authentication and authorisation

- **Store user credentials securely** using techniques such as hashing and salting, and ensure that the credentials are not stored in plaintext.
- **Implement proper authorisation and access control** mechanisms to ensure that users can only access the data and features they are authorised to use.
- **Use industry-standard authentication and authorisation protocols** such as OAuth, OpenID Connect, or Security Assertion Markup Language (SAML) to ensure compatibility with existing authentication and authorisation systems.
- **Use secure communication protocols** such as Hypertext Transfer Protocol Secure (HTTPS) or Secure Sockets Layer (SSL) / Transport Layer Security (TLS) to encrypt the data in transit and prevent interception or tampering.

Funded by the Horizon Europe Framework Programme of the European Union

- **Implement proper session management** techniques such as session timeouts, session invalidation on logout, and protection against session hijacking or fixation attacks.

3.4.4. Protection against malicious attacks

- **Use secure coding practices** to prevent common security vulnerabilities such as SQL injection, cross-site scripting, or buffer overflow attacks.
- **Encrypt all sensitive data** such as user credentials, payment information, or location data, both in transit and at rest.
- **Protect against malware and viruses** by using antivirus software and keeping the software updated with the latest security patches.
- **Protect against denial-of-service attacks** by appropriate architectural design to handle denial-of-service attacks such as network flooding or resource exhaustion attacks.
- **Continuously monitor for security threats** and anomalies, and appropriate measures should be taken to mitigate them.

3.5 GENERAL RECOMMENDATIONS

This section provides general recommendations to be considered for the design and development of the SPIRIT platform, the selection of end-point devices, and the implementation of user applications.

3.5.1 SPIRIT platform

In this section recommendations for two aspects of the SPIRIT platform are provided. The first aspect is about the handling of a growing number of concurrent users with its increase of data. The second is about maintainability, modularity, and extensibility of the SPIRIT platform.

Growing number of users and large amounts of data

- Design the SPIRIT platform to handle increased user activity and data processing requirements. This includes optimising **performance** to ensure smooth operation even when processing large amounts of data.
- Enable the SPIRIT platform to handle increased data storage requirements as the user base grows. This includes implementing **scalable data storage** solutions that can handle large amounts of data efficiently.
- Enable the SPIRIT platform to integrate with **cloud services for scalability**. This includes using cloud-based infrastructure for data storage, processing, and delivery to ensure handling increased traffic and data processing requirements.
- Design the SPIRIT platform with **load balancing capabilities** to distribute server requests across multiple servers. This helps to prevent server overload and ensures the handling of increased traffic and user activity.
- Design the SPIRIT platform with **automated scaling capabilities** to automatically adjust resource allocation based on traffic and user activity.

Funded by the Horizon Europe Framework Programme of the European Union

- Design the SPIRIT platform being **compatible with future updates and new technology**.
- Design the SPIRIT platform to support **scalable orchestration** of computational and networking resources to meet latency and throughput requirements from applications, while taking available resource capacity into account.
- Enable the SPIRIT platform to integrate with **broadcasting / multicasting capabilities** to support efficient transport of common content to multiple users.
- **Test the SPIRIT platform for scalability** to ensure that it can handle increased traffic and user activity without affecting performance. This includes conducting load tests and stress tests to identify any performance issues and optimise the platform's scalability.

Maintainability, modularity and extensibility

- Have a **modular architecture** that allows to add new features or remove existing ones without affecting other parts of the platform. This can be achieved by using well-defined interfaces and separating the platform into components.
- Provide **configurability options** that allow to enable or disable features according to the use case needs.
- Provide **extensible Application Programming Interfaces (APIs)** that allow to create customisations and extensions that can be integrated into applications. This can be achieved by providing well-defined interfaces and documentation that describes how to use them.
- Include proper **error handling** techniques to ensure that applications can gracefully handle errors and exceptions. This includes logging errors and providing clear error messages.
- Use **version control** tools like Git to manage the codebase or components, and track changes.
- Design the SPIRIT platform in such a way that it is **easily testable**.
- **Document the SPIRIT platform** to ensure that developers can easily understand how the platform works and how to extend it. This can be achieved by providing detailed documentation that describes the architecture, APIs and other important aspects of the platform.

3.5.2 End-point device capabilities

The end-point devices are use-case-specific therefore these recommendations are quite general. In some use case the end-point device functions are distributed on more than one device. For instance, GPU processing can be done on the edge cloud, and the local device just combines this output with local information to render the user’s view.

- Have a high resolution, high frame rate, and low latency **camera**. It should also support auto-focus and have a wide field of view to capture a large area. Capture images from the device's camera and process them in real-time with minimal delay.
- The **Inertial Measurement Unit** (IMU) includes sensors such as accelerometers, gyroscopes, and magnetometers. These sensors help in determining the device's position and orientation in real-time. The IMU should have high accuracy, low latency, and low drift.

Funded by the Horizon Europe Framework Programme of the European Union

- A **depth sensor** such as LiDAR or Time-of-Flight (ToF) sensor can improve the accuracy by providing 3D depth information about the environment. It can help in better object tracking, occlusion, and depth perception.
- The **processor** should be powerful enough to handle the computational requirements of the application. It should have multiple cores and a high clock speed for faster processing.
- The **GPU** should be powerful enough to handle the graphics-intensive content. It should have multiple cores and a high clock speed for fast rendering.
- The device should have a **high battery capacity** and support fast charging to ensure that the battery lasts for a reasonable amount of time.

3.5.3 User interface design

- Have a **clear and intuitive navigation** system that allows users to easily access all features and functions.
- Have a **responsive and interactive** user interface that provides a seamless experience for users.
- Follow a **consistent design** language that is in line with the overall brand and user experience.
- Make **use of visual cues** such as arrows, pointers, or highlights to guide users to relevant elements or actions.
- Consider the **capabilities and limitations** of the device it is being used on. For example, the user interface should be designed to work with the device's screen size and resolution, as well as its processing power and battery life.
- Have an **intuitive user interface** that is easy to navigate and understand, with clear instructions and visual cues.
- Provide a **smooth user experience** with minimal lag or delays, allowing users to interact with XR elements seamlessly.
- Provide **realistic interactions** with XR elements, such as realistic physics or movement, to create a good immersive experience.
- Provide **haptic feedback**, such as vibration or force feedback, to enhance the user experience and provide additional sensory input.
- Incorporate **gesture-based interactions**, such as swiping or tapping, to allow users to interact in a natural and intuitive way.
- Incorporate **voice-based interactions**, such as voice commands or voice recognition, to allow users to interact with AR elements hands-free.
- Allow users to **customise** their **settings**, such as the sensitivity of gesture-based interactions or the level of haptic feedback.
- Provide **user feedback**, such as visual or auditory cues, to let users know when an action has been successfully completed or if there are any errors.
- Have **effective error handling**, with clear and concise error messages that provide users with guidance on how to correct any issues.

Funded by the Horizon Europe Framework Programme of the European Union

- Security: **Validate all user input** to prevent input validation attacks such as command injection or cross-site scripting attacks.
- **Stability**: The application should be stable and reliable, and it should not crash or freeze during use. The application should be able to handle unexpected inputs and errors without interrupting the user's experience.

Funded by the Horizon Europe Framework Programme of the European Union

4 THE SPIRIT ARCHITECTURE

In this section we first present the evolved design of the general SPIRIT architecture in Section 4.1, then we describe the security architecture in Section 4.2.

4.1 UPDATED SPIRIT PLATFORM ARCHITECTURE AND INTERFACES

4.1.1 Platform architecture specification

In D2.1 we presented the initial version of the SPIRIT architecture with basic building blocks for supporting specific project-defined use cases. In this deliverable we present the evolved architecture aiming to facilitate the specification of function components and interfaces in the context of both project-defined use cases and innovations. In this deliverable we mainly articulate the components in the context of use cases and in D3.2 (SPIRIT Consortium, Innovation Platform Enablers (Second Version), 2024) the diagram will also be used for explaining specific innovations in more details.

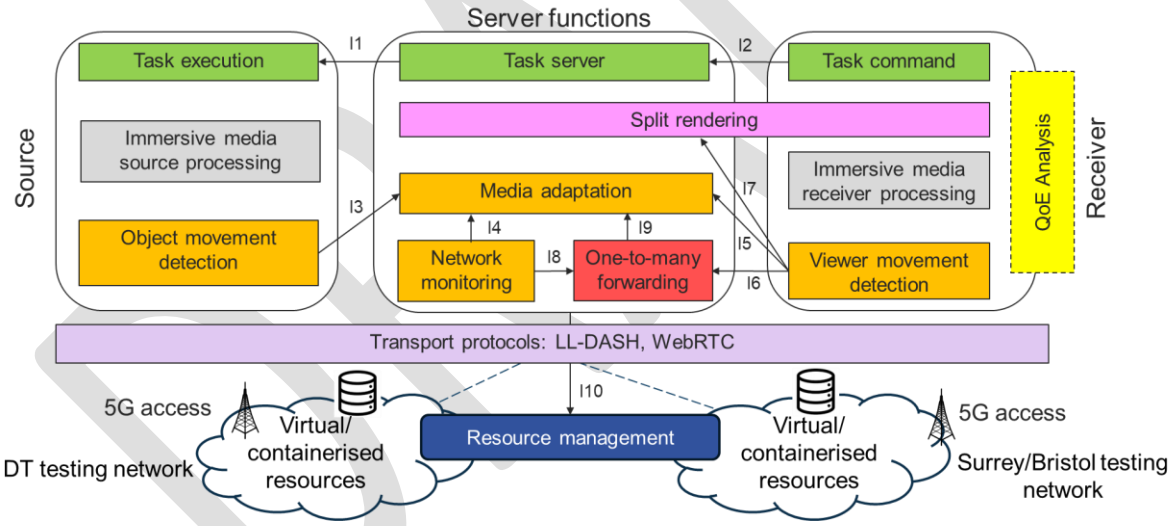

FIGURE 7 THE SPIRIT PLATFORM ARCHITECTURE

Funded by the Horizon Europe Framework Programme of the European Union

As shown in

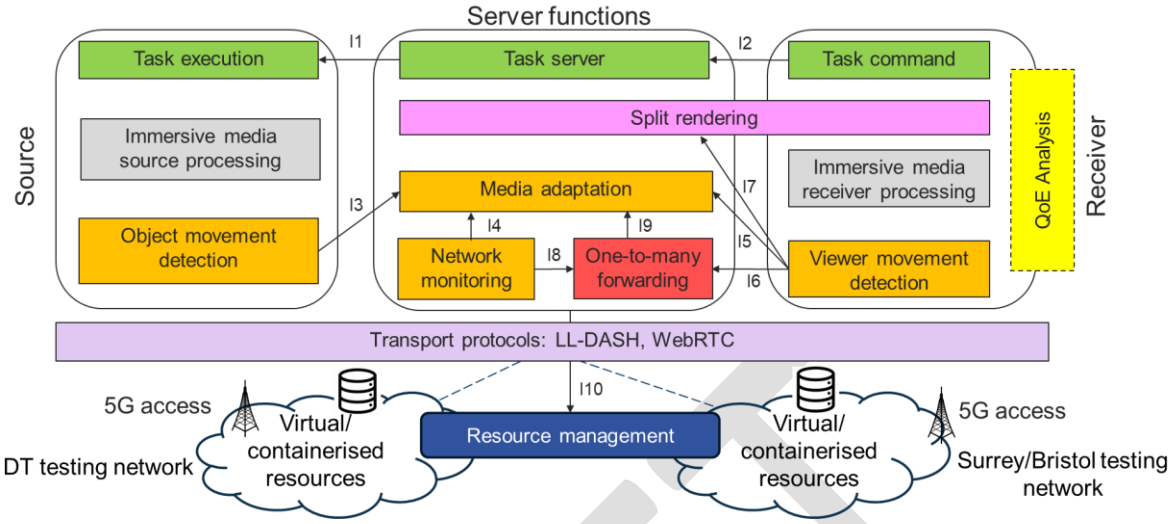

Figure 7, the evolved architecture inherits from the previous version the distributed testing infrastructure between the two sites at DT and Surrey. As shown in D4.2 (SPIRIT Consortium, SPIRIT Platform (Second Version), 2024) both sites have been provisioned with 5G access and IT resources that are virtualised / containerised for supporting the operation of a wide variety of use case applications. The upper part of the architecture focuses on the organisation of function components and their interactions across three different entities:

- **Source client**: The term *source* is defined as the object being live captured at the camera side and whose avatar/hologram is transmitted through the SPIRIT platform towards the viewer side. In the Robot use case, it also covers the robot role – on one hand it is viewed in real-time by the viewer side and on the other hand it can receive specific task commands from the viewer in order to execute specific operations. The *source client* is defined as a group of function components on the source side which are typically hosted by a dedicated edge computing node.
- **Receiver client**: The *receiver* is defined to be the end user who views the object's avatar or hologram in real-time. In the Robot use case, the receiver is also supposed to issue task commands to control the robot on the source side. The *receiver client* is defined as a group of function components on the receiver side which are typically hosted by a user equipment such as head mounted devices (HMDs).
- **Server**: The *server* node between the source and the receiver clients typically hosts a set of centralised function components to facilitate the end-to-end operation of telepresence applications. Different use cases may require specific sets of function components involved at the server side, and it can be also the case that some use case applications have a direct communication between the source client and the receiver client without involving the role of the server.

Funded by the Horizon Europe Framework Programme of the European Union

We now elaborate on the platform architecture shown in

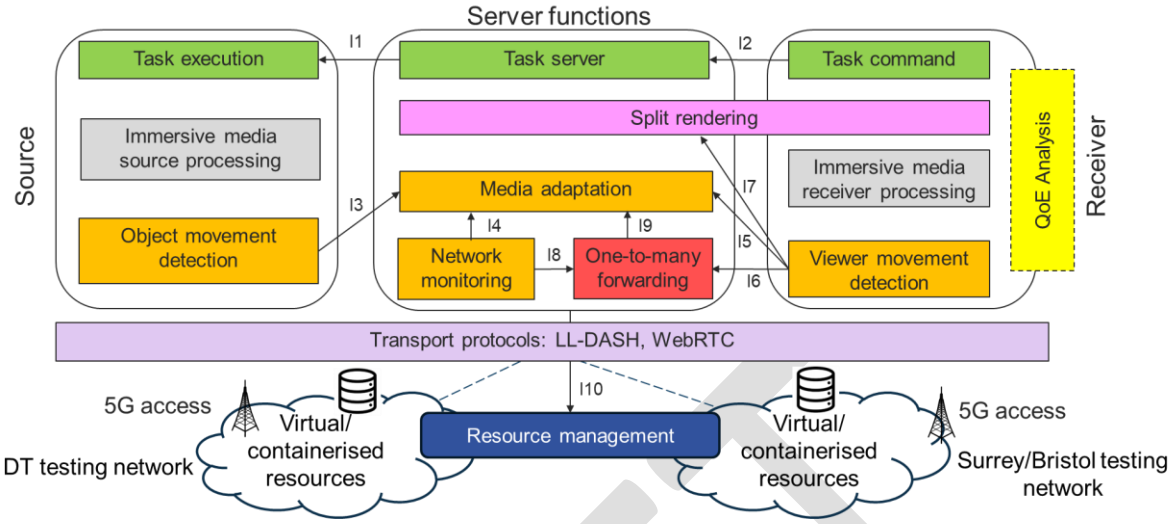

Figure 7 in the context of SPIRIT use cases and innovations. It is worth noting that this architecture is to illustrate the organisation of software functions (including their interactions) rather than showing the actual data flow path. Depending on specific use case scenarios, the content data flow from the source client may or may not go through the server in the middle before reaching the receiver client, but this is not shown in the figure. Secondly, the SPIRIT use cases consider two different types of models, namely avatar based and holographic based, each with their specific encoding, decoding and rendering techniques. Again, this architecture diagram is not supposed to be specific to any models in order to retain its generality.

On both ends of the architecture there are the functions of immersive media source and receiver processing including encoding and decoding, which are generic ones regardless of specific models in different use cases. One innovation on this is **parallel media encoding and decoding**, which functions on both ends in order to reduce the end-to-end delay. Several solutions can be adopted to realise this objective. One example is based on multi-description coding (MDC), where all points belonging to a captured point cloud frame are divided in distinct subsets or descriptions. Each description is compressed in parallel, resulting in lower execution delays than when the frame would be compressed as a whole. All descriptions can then be forwarded to a central server (e.g., a WebRTC MCU or SFU), which delivers only the relevant descriptions to the receiving clients. Making use of this central server component, support for **one-to-many and many-to-many communication** can be foreseen. The architecture also includes the organisation of relevant functions for the innovation of supporting **intelligent media adaptation** operations in dynamic environments according to different factors. Specifically, in addition to taking into account the network resource availability (data rate throughput) from the network side, the adaptation decision-making also considers the context information from both the receiver and the source side. In the former case, if the viewer with the HMD moves closer to the object, the higher resolution is required. In the latter case, if the object on the source side is moving slowly or even is in stationary status, then the frame rate can be reduced without significantly affecting viewer's QoE. In the case of MDC-based point cloud delivery, more than one description can be delivered to increase the visual quality if the available bandwidth is significantly high. The ultimate purpose is to strike an optimised balance between media resolution, quality and frame rate by taking into account different factors in relevant use cases. The innovation of **split rendering** between the server and the receiver client sides is also captured in the architecture. The key purpose is to facilitate flexible sharing of rendering process load between the server and the receiver client such as HMDs. For instance, in the Avatar based use case, this approach is used to enable the transmission

Funded by the Horizon Europe Framework Programme of the European Union

of video and audio of an avatar with a significant reduction in the required bandwidth. By carrying out the majority of the rendering process on a server (that can access a much wider variety of resources, such as GPUs with hardware acceleration), the time that this task requires is much lower, since it prevents the server from sending the whole geometry and texture of the 3D objects. Besides this, it provides a much better flexibility for the rendering process, since different technologies can be used without the clients needing to be aware of it. These clients just need to implement simple algorithms to finish the rendering process, such as fast background removals. On the other hand, the additional overhead required for the synchronisation of the whole architecture consists only of lightweight messages containing the position of the object within the scene, which does not add significant load to the communication. It is worth mentioning that, although we illustrate the innovation of split rendering based on the avatar model, it can be certainly generalised to the holographic model as well.

Concerning the support of human-machine (robot) interaction feature, the Robotics server acts as the central brain for the autonomous mobile robots (AMRs), managing communication with both the robots and users. It receives critical data, such as laser scans, safety zones, odometry, and battery status, through a communication interface (ComLink) from an AMR Client running on each robot. This ComLink interface operates independently from our Robot Control that runs on the server, ensuring continuous data communication with the broader framework using multiple protocols like ROS, MQTT, HTTP, and our custom TCP-based ComLink. The server's Robot Control processes this data to handle complex functions like perception, localisation, mapping, behavior control, navigation planning and docking. In the end our robot control only communicates essential drive commands and specific robot actions, like arm or lift controls, back to the AMR. Moreover, the Robotics server provides a user interface that allows for manual robot control, issuing autonomous navigation commands, and real-time monitoring through a map or video streams.

Resource management on the local testbed infrastructures ensures proper allocation of computational and networking resources to support efficient deployment of immersive applications and it requires up-to-date knowledge about the IT computation and network resource conditions. Computational resources are managed by a container orchestrator (such as open source Kubernetes, K8s) to allow for automated deployment and scaling of the containerised application components. Optimal resource scheduling can be based on the default K8s scheduler (dealing with RAM and CPU metrics only) or using the Diktyo scheduler, a SPIRIT innovation that also takes network characteristics and application dependencies into account to meet the stringent low-latency and high-throughput requirements of immersive applications.

Although **QoE analysis** is deemed to be an offline process, for the sake of completion we include this item in the architecture diagram. Online (in-the-loop) QoE analysis or estimation is a challenge even in 2D video streaming, even more so in immersive environments for which QoE analysis is just evolving. This process is thus done offline in SPIRIT, either by intricate subjective QoE studies to derive QoE guidelines (as done for and documented in D4.1 and D4.2 for point-cloud content) or by objective QoE estimation using representative 2D videos rendered from presentations to the end users and an established objective QoE model (P.1203, as done for the SPIRIT use cases and documented in D4.2).

Detailed information about the mentioned innovations will be presented in D3.2.

Funded by the Horizon Europe Framework Programme of the European Union

4.1.2 Platform interface specification

We now provide more detailed descriptions of the interfaces in the architecture diagram to facilitate the interactions between related functions. For the sake of convenience, they are simply named based on number indexes I(nterface)1 to I10 as listed below. For those interfaces requiring more detailed elaborations, we also have dedicated diagrams in Figure 8, Figure 9 and Figure 10 to facilitate understanding.

- **I1**: The interface utilises the Robot Operating System (ROS) for communication between the robot and the server. ROS employs a publish/subscribe model for efficient message passing. The server derives cmd_vel messages, which are one of ROS's standard message types, from inputs received through either manual or autonomous navigation inputs. Ultimately, only movement-related instructions are sent to the mobile robot to keep the interface as generic as possible for supporting multiple robots.
- **I2**: The interface offers multiple modes for controlling a mobile robot, including manual steering and autonomous navigation options. Through the UI, users can view a live video feed from the robot's camera and issue real-time drive commands. In manual mode, users control the robot by sending velocity commands to adjust its linear speed (forward/backward) and angular velocity (rotation), along with a deviceID that specifies which robot is being controlled. In autonomous modes, users can select a Point of Interest (POI) as a destination. The deviceID and POI are sent to the server, where the optimal path for the robot is calculated to reach the destination. The UI communicates with the server using RESTful APIs to send velocity commands or POI's for autonomous navigations and receive status updates, such as battery levels and connection state.

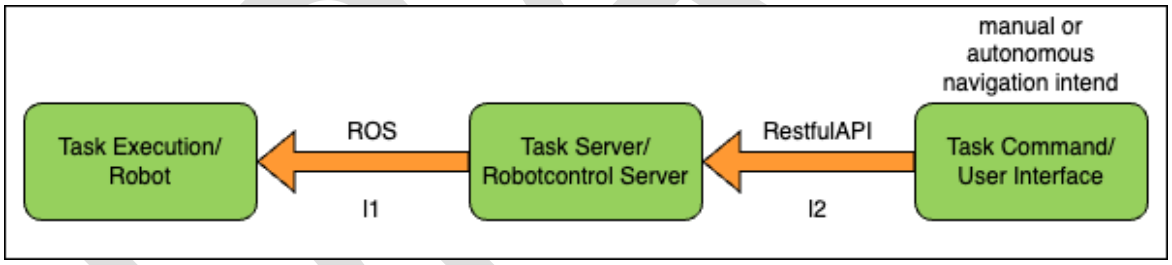

FIGURE 8 THE INTERFACE INVOLVED IN THE ROBOTIC CONTROL USE CASE

- **I3:** This interface is used to allow the source client to feed the server the real-time information about object movement patterns in order for the latter to make media adaptation decisions. This is typically through the real-time capturing of the source object skeleton data by the cameras and feeding the data to the media adaptation function on the server side. This interface is currently implemented based on the Microsoft Azure DK cameras with 32 identifiable joints in the 3D X/Y/Z space which can help to enable the detection of movement patterns of the object.
- **I4:** This interface is to facilitate the feeding the knowledge of the server about the current network conditions in order to make appropriate decisions on dynamic media adaptation. Currently this interface is implemented with RESTful APIs to transmit the information on real-time network conditions including delay, packet loss and throughput performances. Such information is leveraged for comprehensive decision-making about media adaptations in both one-to-one and one-to-many telepresence scenarios.

Funded by the Horizon Europe Framework Programme of the European Union

In the latter case, the decisions are made according to the monitored network conditions on a per-receiver basis.

- **I5:** The viewer behaviour handling facet addresses the problem regarding uncertainties associated with unpredictable viewer behaviour that may influence the decision-making on media adaptations. To effectively enable this facet, this interface is responsible for transmitting real-time viewer information such as location data obtained from HMDs like the Microsoft HoloLens 2 to determine if viewer intent has been expressed relative to the determined intent threshold.

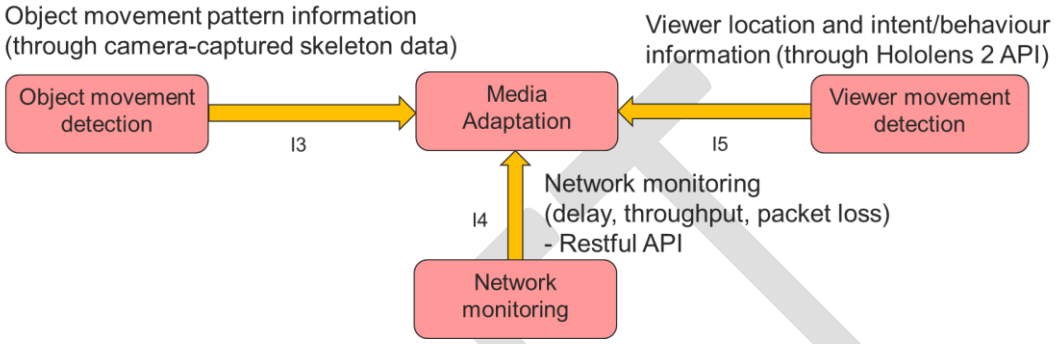

FIGURE 9 INTERFACES INVOLVED IN MULTI-DIMENSIONAL MEDIA ADAPTATION

- **I6**: Similar to I5, this interface is used to communicate information on the user's location and viewport, obtained from HMDs such as the Meta Quest 2 and 3. A cartesian coordinate system is used to indicate the user's location, while orientation quaternions are used to reflect the rotation of the user's head. The angular FoV of the considered HMD is passed along as well.
- **I7**: The interface is used as a communication channel from the receiver client to the server containing information about the current positioning of the receiver device (viewpoint) and of the virtual objects present in the scene. This information, that must be structured accordingly, is then used by the server to synchronise the location of the virtual different objects and cameras, allowing the rendering of the appropriate views and keeping a central controller to manage all the elements of the scene.
- **I8**: UDP-based solutions inherently lack flow control and bandwidth estimation mechanisms. IMEC's WebRTC solution for volumetric video delivery makes use of Google's congestion controller to estimate the available bandwidth on the application layer. Estimations can be retrieved through this interface and can be used to adapt the quality of forwarded video tracks.
- **I9**: This interface is used to communicate decisions by the quality adaptation algorithm, implemented in the WebRTC MCU or SFU. This algorithm requires as input (i) information on the user's location and viewport, (ii) the estimated bandwidth available to the client, (iii) the location of other users in the scene and (iv) the available quality representations and their respective bitrates. The output is a list containing the selected quality representations for each user.
- **I10**: The interface to the resource management process on the local testbeds is accessible through the regular K8s control plane (K8s HTTP API server). It can be used to report the monitored computational and network resource conditions to be fed into the resource management function.

Funded by the Horizon Europe Framework Programme of the European Union

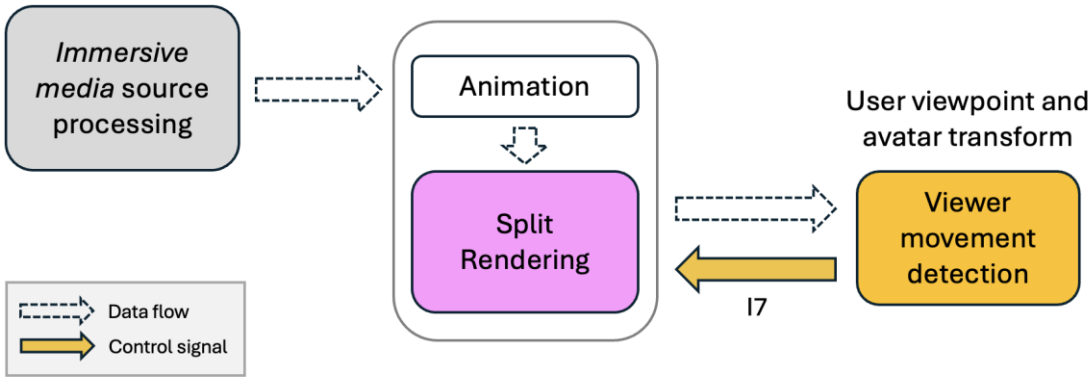

FIGURE 10 INTERFACES INVOLVED IN THE AVATAR USE CASE

4.2 SECURITY ARCHITECTURE

4.2.1 Security challenges

4.2.1.1 Cloud-based applications and frameworks

More and more companies move workloads from on-premise systems in their own datacentres to shared resources provided by public cloud providers (such as Amazon AWS, Microsoft Azure, Google Cloud and others). The advantages of this approach are – among others – cost reduction and more flexible resource allocation.

However, this move to cloud resources creates some security headaches:

- While data is nowadays normally encrypted while in transit, it must be decrypted in the cloud to allow processing.
- This leaves possibly sensitive data open to theft and government surveillance.
- Most cloud providers are headquartered outside of the European Union, therefore achieving regulatory compliance (GDPR) can be difficult or impossible to achieve for European data owners.

These considerations are especially pertinent for cloud-based telepresence frameworks and applications developed in the SPIRIT project, as these applications naturally involve the processing of personal data.

4.2.1.2 Identity management in telepresence applications

Identity management in telepresence applications is a problem because it is difficult to ensure that the person controlling the telepresence device is who they claim to be. This can lead to issues such as unauthorised access to sensitive information or physical spaces, as well as impersonation and deception. Additionally, telepresence systems often involve multiple parties and different levels of access, making it challenging to manage and control access to different resources. Ensuring secure and reliable identity management in telepresence applications

Funded by the Horizon Europe Framework Programme of the European Union

requires the use of robust authentication and authorisation methods, as well as ongoing monitoring and auditing of access.

4.2.1.3 Attack surface of distributed systems

The attack surface of a distributed system – and a telepresence system is distributed by nature – is large because a distributed system typically consists of multiple interconnected components that are spread across different locations and networks. This can include servers, clients, network devices, and other devices that are connected to the system.

Additionally, distributed systems often involve communication between different components, which can involve multiple protocols and ports. This increases the potential for vulnerabilities and attack vectors that can be exploited by attackers.

Furthermore, distributed systems often operate over the public Internet, which can make them more susceptible to attacks such as denial of service (DoS) and distributed denial of service (DDoS) attacks.

Additionally, distributed systems usually have more complex configurations and interconnections than centralised systems, adding more complexity to the system and making it harder to secure and monitor.

Finally, distributed systems often rely on third-party services, such as cloud providers, which can introduce additional security risks.

Overall, the large attack surface of distributed systems is due to the complexity and interconnectedness of the system, the multiple communication channels, and the use of public networks, which can make it more difficult to detect and defend against attacks.

4.2.2 Architectural elements

4.2.2.1 Confidential computing

Confidential computing offers a possible solution to this conflict of goals as laid out in the previous section, by offering enhanced security of data processing in the cloud while keeping the advantages of moving to shared resources.

When workloads are moved from on-premise deployments to the cloud, new attack surfaces are exposed that did not exist in the own data centre: The cloud trusted computing base (TCB) additionally encompasses the management systems of the cloud provider, its employees, as well as government agencies from the jurisdiction of the cloud provider.

Similar security concerns exist when deploying workloads to edge devices: Here the devices may also be under the control of an edge provider (e.g., a telecom operator) or they may be deployed in exposed locations where they may be physically compromised by bad actors (e.g., road-side units).

In this context, we introduce the concept of secure remote computation (also referred to as confidential computing in this document) as the problem of executing software on a remote computer owned and maintained by an untrusted party, with some integrity and confidentiality guarantees.

Funded by the Horizon Europe Framework Programme of the European Union

A possible solution would be to end-to-end encrypt data while traversing cloud systems. This, however, would not allow for any meaningful and performant processing of data in the cloud. New mathematical approaches such as homomorphic encryption allow some limited processing on encrypted data but incur an extreme overhead and are only suitable for very limited application scenarios. Basically, the processing of encrypted data is a field of active research not ready for wide-spread commercial application. In the general setting, secure remote computation is an unsolved problem. Fully Homomorphic Encryption solves the problem for a limited family of computations but has an impractical performance overhead [12].

Figure 11 presents a new approach: end-to-end security, where data in transit is encrypted and only provably trustworthy cloud applications can break up this encryption to process and store the confidential data. To exclude the cloud provider itself from the 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.

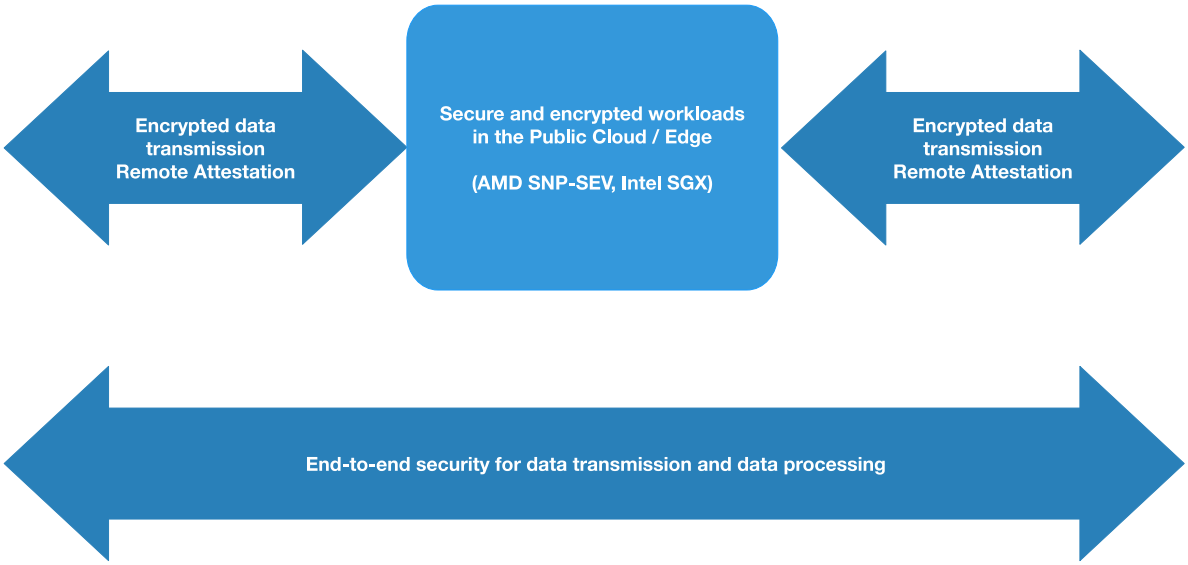

FIGURE 11: END-TO-END SECURITY

4.2.2.2 Identity management

Identity management in the context of telepresence refers to the process of ensuring the authenticity and integrity of the identity of users interacting through telepresence systems. This can include both human users and robotic avatars.

One approach to identity management in telepresence is to use biometric authentication methods such as facial recognition, fingerprint scanning, or voice recognition. These methods can help to ensure that the person operating the telepresence system is who they claim to be.

Another approach is to use digital certificates and encryption to secure the communication between the telepresence system and the remote user. This can help to prevent unauthorised access to the system and protect the privacy of the user.

Overall, a combination of different identity management approaches can be used to ensure the security and privacy of telepresence systems, and to prevent unauthorised access to the system.

Funded by the Horizon Europe Framework Programme of the European Union

4.2.2.3 Intrusion detection systems

IDSs play an important role in ensuring the security of telepresence systems. These systems monitor network traffic and identify suspicious activity that may indicate an attempted intrusion or attack.

One approach to intrusion detection in telepresence systems is to use signature-based detection methods. This approach involves comparing network traffic against a predefined set of known attack signatures. If a match is found, the system can take appropriate action to block the attack.

Another approach is to use behaviour-based detection methods, which involve monitoring network traffic for patterns of activity that are unusual or indicative of an attack. This approach is useful for detecting new, unknown attacks that may not have a known signature.

A third approach is to use machine learning-based detection methods, which can learn and then adapt the normal behaviour of the system and detect anomalies that may indicate an attack. These methods can also be combined with other techniques to improve the accuracy of intrusion detection.

Overall, intrusion detection systems are essential to ensure the security of telepresence systems, and a combination of different intrusion detection techniques can be used to provide comprehensive security coverage.

Funded by the Horizon Europe Framework Programme of the European Union

5 CONCLUSIONS

Based on D2.1, this deliverable provides the updated version of the SPIRIT use cases with expanded scope of application and system requirements, as well the enhancement of the overall SPIRT architecture and interfaces. The use cases identified by the project team cover a wide variety of remote telepresence application scenarios, including both live human-to-human communications (with and without Avatars) and real-time human-to-machine interactions. In addition, this deliverable also outlines the broad scope of future desirable use case scenarios in different vertical sectors, which can be used as guidelines for attracting the participation of new application development by third-party participants (Open Calls). Based on these use cases, the project team have also derived a range of requirements and recommendations in the areas of network and transport support, as well as on the application and security sides. An enhanced version of the overall SPIRIT architecture is also documented in this deliverable, encompassing key architecture components of the SPIRIT platform. Such an architecture is supposed to illustrate the organisation of relevant function components and the interfaces between them in the context of the SPIRIT use cases and innovations.

Funded by the Horizon Europe Framework Programme of the European Union

6 REFERENCES

[1] M. Kowalski, J. Naruniec, M. Daniluk, “Livescan3D: A Fast and Inexpensive 3D Data Acquisition System for Multiple Kinect v2 Sensors”, Proc. IEEE International Conference on 3D Vision, 2015

[2] S. Anmulwar, N. Wang, V. S. H. Huynh, S. Bryant, J. Yang, R. R. Tafazolli, “HoloSync: Frame Synchronisation for Multi-Source Holographic Teleportation Applications”, IEEE Transactions on Multimedia, vol. 25, pp. 6245-6257, 2023

[3] ZSTD compression, Zstandard - Real-time data compression algorithm, available at: http://facebook.github.io/zstd/

[4] Azure Kinect Body Tracking SDK, available at: https://learn.microsoft.com/en-us/azure/kinect-dk/body-sdk-download

[5] Microsoft MRTK3 Mixed Reality Toolkit 3 Developer Documentation, available at: https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk3-overview/

[6] J. Son et al. "Split Rendering for Mixed Reality: Interactive Volumetric Video in Action", Proc. SIGGRAPH Asia 2020 XR, 2020

[7] Ericsson Consumer and Market Insight report, Five ways to a better 5G, available at: https://www.ericsson.com/en/reports-and-papers/consumerlab/reports/five-ways-to-a-better-5g

[8] Ericsson Consumer and Market Insight report, 5G consumer potential, available at: https://www.ericsson.com/en/reports-and-papers/consumerlab/reports/5g-consumer-potential

[9] Ericsson Consumer and Market Insight report, The dematerialized office, available at: https://www.ericsson.com/en/reports-and-papers/industrylab/reports/the-dematerialized-office

[10] SPIRIT Consortium, “Innovation Platform Enablers (First Version),” Project "Scalable Platform for Innovations on Real-time Immersive Telepresence", EC grant agreement 101070672, 2023.

[11] SPIRIT Consortium, “SPIRIT Platform (First Version),” Project "Scalable Platform for Innovations on Real-time Immersive Telepresence", EC grant agreement 101070672, 2023.

[12] El Makkaoui, Khalid, Abderrahim Beni-Hssane, and Abdellah Ezzati. "Cloud-ElGamal: An efficient homomorphic encryption scheme." 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM). IEEE, 2016.

Funded by the Horizon Europe Framework Programme of the European Union

7 APPENDIX: SECURITY REQUIREMENTS KPIS

In order to be able to test, evaluate and validate security requirements, the responsible partners will be requested to fill in the following security questionnaire, evaluating their software according to the evaluation methodology. The evaluation result cells should document concrete measures taken to fulfil the security requirement (not only “yes” or “no”).

The resulting security evaluations will be published in deliverable D4.3.

Requirement	**Evaluation methodology**	**KPIs**	**Evaluation result**
Strong encryption of all communication paths in order to protect data-in-transit.	Analyse design and implementation of the software component.	Yes/no Encryption algorithms and key lengths (AES minimum 256 bits, RSA minimum 2048 bits, other mechanisms accordingly)	
Use of **confidential computing** mechanisms to protect data-in-use (specifically in the public cloud or exposed edge computing facilities).	Analyse design and implementation of the software component.	Yes/no	
Implementation of an **intrusion detection system** (IDS) to protect the telepresence system.	Analyse design and implementation of the software component.	Yes/no	
Secure identification of participants using identity management solutions tailored to the specific properties of telepresence systems.	Analyse design and implementation of the software component.	Yes/no	

Funded by the Horizon Europe Framework Programme of the European Union

Inform users about the types of data collected, such as location data, camera feed, or user input, and how the data will be used.	Inspect documentation and user interface.	Yes/no	
Obtain user consent before collecting any personal data or tracking user behaviour. The consent should be explicit, informed, and revocable at any time.	Inspect documentation and user interface.	Yes/no	
Store user data securely and protect it from unauthorized access, modification, or disclosure. The data should be encrypted at rest.	Analyse design and implementation of the software component.	Yes/no Encryption algorithms and key lengths (AES minimum 256 bits, RSA minimum 2048 bits, other mechanisms accordingly)	
Ensure the integrity of user data, including tracking data, user inputs, and server communications. The application should use encryption and other security measures to protect user data from unauthorized access or tampering.	Analyse design and implementation of the software component.	Yes/no Encryption algorithms and key lengths (AES minimum 256 bits, RSA minimum 2048 bits, other mechanisms accordingly)	
Comply with data protection regulations, such as the **General Data Protection Regulation (GDPR)**,	Inspect documentation and user interface.	Yes/no	

Funded by the Horizon Europe Framework Programme of the European Union

and other regulations, depending on the type of data being collected.			
Allow users to delete their data and provide clear instructions on how to do so. The application should also ensure that the deleted data is completely removed from all backups and systems.	Inspect documentation and user interface.	Yes/no	
Store user credentials securely using techniques such as hashing and salting, and ensure that the credentials are not stored in plaintext.	Analyse design and implementation of the software component.	Yes/no Use secure hash mechanisms such as SHA256 or equivalent	
Implement proper authorisation and access control mechanisms to ensure that users can only access the data and features they are authorised to use.	Analyse design and implementation of the software component.	Yes/no	
Use industry-standard authentication and authorisation protocols such as OAuth, OpenID Connect, or Security Assertion Markup Language (SAML) to ensure compatibility with existing authentication and	Analyse design and implementation of the software component.	Yes/no	

Funded by the Horizon Europe Framework Programme of the European Union

authorisation systems.			
Use secure communication protocols such as Hypertext Transfer Protocol Secure (HTTPS) or Secure Sockets Layer (SSL) / Transport Layer Security (TLS) to encrypt the data in transit and prevent interception or tampering.	Analyse design and implementation of the software component.	Yes/no Encryption algorithms and key lengths (AES minimum 256 bits, RSA minimum 2048 bits, other mechanisms accordingly) TLS level minimum 1.2	
Implement proper session management techniques such as session timeouts, session invalidation on logout, and protection against session hijacking or fixation attacks.	Analyse design and implementation of the software component.	Yes/no	
Use secure coding practices to prevent common security vulnerabilities such as SQL injection, cross-site scripting, or buffer overflow attacks.	Inspect software development environment.	Yes/no At least one tool to support secure software development (such as SonarQube) must be used. It's usage must be enforced.	
Encrypt all sensitive data such as user credentials, payment information, or location data, both in transit and at rest.	Analyse design and implementation of the software component.	Yes/no Encryption algorithms and key lengths (AES minimum 256 bits, RSA minimum 2048 bits, other	

Funded by the Horizon Europe Framework Programme of the European Union

		mechanisms accordingly)	
Protect against malware and viruses by using antivirus software and keeping the software updated with the latest security patches.	Analyse design and implementation of the software component.	Yes/no	
Protect against denial-of-service attacks by appropriate architectural design to handle denial-of-service attacks such as network flooding or resource exhaustion attacks.	Analyse design and implementation of the software component.	Yes/no	
Continuously monitor for security threats and anomalies, and appropriate measures should be taken to mitigate them.	Analyse design and implementation of the software component.	Yes/no	

Funded by the Horizon Europe Framework Programme of the European Union