D3.2 INNOVATION PLATFORM ENABLERS (SECOND VERSION)

Revision: v.1.0

Work package	WP3
Task	Task 3.1, 3.2, 3.3 and 3.4
Due date	30/09/2024
Submission date	30/09/2024
Deliverable lead	Ericsson
Version	1.0
Authors	Nick Turay (EDD), Jeroen van der Hooft, José Santos, Time Wauters (IMEC), Ali El Essaili - Editor (EDD), Christoph Stielow (TSI), Peter Hofmann, Wolfgang Paier, Wieland Morgenstern (DT-Sec), Sergio Tejeda Pastor (Fraunhofer), Hermann Hellwagner and Shivi Vats (UNI-KLU), Peng Qian (UOS), Carl Udora (UoS)
Reviewers	Sergio Tejeda Pastor (HHI)
Abstract	This report presents the second version of the innovation platform enablers focusing on immersive telepresence. The innovations are intended to extend the SPIRIT platform by bringing innovations that improve the scalability and user experience in telepresence solutions.
Keywords	3D, 5G, AR, MR, Avatar, Holographic Communication, Immersive Telepresence, LL-DASH, Many-To-Many Conferencing, Network Aware Scheduling, One-To-Many Conferencing, One-To-One Conferencing, Real-Time Streaming, Resource Management, Scalability, Security, Split Rendering, User-Intent Driven Adaptation, WebRTC

Document Revision History

Version	Date	Description of change	List of contributors
V1.0	30/09/2024	Second published version	Nick Turay, Jeroen van der Hooft, José Santos, Time Wauters (IMEC), Ali El Essaili-Editor (EDD), Christoph Stielow (TSI), Peter Hofmann, Wolfgang Paier, Wieland Morgenstern (DT-Sec), Sergio Tejeda Pastor (Fraunhofer), Hermann Hellwagner and Shivi Vats (UNI-KLU), Peng Qian (UOS), Carl Udora (UoS)

DISCLAIMER

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

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

COPYRIGHT NOTICE

© 2022 - 2025 SPIRIT Consortium

Project co-funded by the European Commission in the Horizon Europe Programme		
Nature of the deliverable:	**to specify R, DEM, DEC, DATA, DMP, ETHICS, SECURITY, OTHER***	
Dissemination Level		
PU	*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.

EXECUTIVE SUMMARY

This deliverable for Work Package 3 (WP3) presents the second version of the innovation platform enablers for the SPIRIT project focusing on immersive telepresence. More specifically, the innovations enhance the functionality of initial SPIRIT platforms presented in D3.1 [1] and are aligned with evolved architecture and use-cases outlined in D2.2 [2]. Different innovations are described in this report including scalability, one-to-many split rendering, video-driven avatar animation, and research investigations on low latency DASH transport for immersive media, multi-dimensional media adaptation, and security and software enhancements.

Furthermore, several of the innovations developed in the first version have already been incorporated into the application platforms at 5G network testbeds, and they are made available for the first Open Call partners. These innovations tackle important aspects of the SPIRIT platform such as end-to-end holographic communication, resource management, scalability, split rendering as well as security with the goal to build a more sophisticated SPIRIT platform.

For *D3.2 – Innovation Platform Enablers (Second Version),* the focus is on extending the initial innovations. The findings of this report are aimed to be made available to Open Call participants as part of the second call.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

3D	Three-Dimensional
5G	Fifth Generation
5GIC	5G Innovation Centre
6DoF	Six Degrees of Freedom
ACM	Authenticated Code Module
ACME	Automatic Certificate Management Environment
API	Application Programming Interface
AR	Augmented Reality
ASA	Adaptive Security Appliance
AWS	Amazon Web Services
BIOS	Basic Input/Output System
CAS	Configuration and Attestation Service
CDN	Content Delivery Network
CI/CD	Continuous Integration & Delivery
CMAF	Common Media Application Framework
CNN	Convolutional Neural Network
CO	Confidential
COS	Container-Optimised OS
CPU	Central Processing Unit
CSP	Cloud Service Provider
CTE	Chunked Transfer Encoding
DASH	Dynamic Adaptive Streaming over HTTP
DCAP	Datacentre Attestation Primitives
DRAM	Dynamic Random Access Memory
DT	Deutsche Telekom
EDD	Ericsson

ES	Encrypted State
FoV	Field of View
GCC	Google Congestion Control
HAS	HTTP Adaptive Streaming
HHI	Heinrich-Hertz Institute
HLS	HTTP Live Streaming
HTTP	HyperText Transfer Protocol
ICE	Interactive Connectivity Establishment
ID	Identifier
IETF	Internet Engineering Task Force
I/O	Input/output
IP	Internet Protocol
JSON	JavaScript Object Notation
JWT	JSON Web Token
K8s	Kubernetes
KVM	Kernel-based Virtual Machine
L4S	Low Latency, Low Loss, Scalable Throughput
LAS	Local Attestation Service
LL-DASH	Low-Latency DASH
MCU	Multipoint Control Unit
MPD	Media Presentation Description
MPEG	Moving Picture Experts Group
MR	Mixed Reality
NALU	Network Abstraction Layer Unit
NAT	Network Address Translation
NUC	Next Unit of Computing
OS	Operating System
OSI	Open System Interconnection

P2P	Peer-To-Peer
PAL	Platform Adaptation Layer
PCE	Provisioning Certification Enclave
PCK	Provisioning Certification Key
PSP	Platform Secure Processor
QoE	Quality of Experience
RAM	Random-Access Memory
RIC	RAN Intelligent Controller
RSA	Rivest–Shamir–Adleman
RT	Real-Time
SDK	Software Development Kit
SDN	Software Defined Networking
SDP	Signal Description Protocol
SEAM	Secure-Arbitration Mode
SECT	Security in Telecommunications
SEV	Secure Encrypted Virtualization
SFC	Service Function Chain
SFU	Selective Forwarding Unit
SGX	Software Guard Extensions
SMLP	Skinned Multi-Person Linear Model
SMM	System Management Mode
SNP	Secure Nested Pages
STUN	Session Traversal Utilities for NAT
TCB	Trusted Computing Base
TCP	Transmission Control Protocol
TD	Trust Domain
TDX	Trust Domain Extensions
TEE	Trusted Execution Environment

TPM	Trusted Platform Module
TURN	Traversal Using Relays around NAT
TXT	Trusted Execution Technology
UDP	User Datagram Protocol
UEFI	Unified Extensible Firmware Interface
UNI-KLU	University Klagenfurt
UoS	University of Surrey
VM	Virtual Machine
VMM	Virtual Machine Manager
VPN	Virtual Private Network
VR	Virtual Reality
W3C	World Wide Web Consortium
WebRTC	Web Real-Time Communication
Wi-Fi	Wireless Fidelity
WP3	Work Package 3
XR	Extended Reality

1 INTRODUCTION

The goal of WP3 is to carry out innovations across all key aspects related to the empowerment of collaborative telepresence services with optimised user experiences in different scenarios, and the support of large-scale operations. More specifically, these innovations consider content, application, transport, network and security aspects.

At the beginning of the project, this initiative aims to enable immersive telepresence applications in a collaborative framework by extending implemented telepresence platforms and network testbeds from the project consortium with more innovations. The innovations aim to provide practical scalability of telepresence content delivery while upholding robust Quality of Experience (QoE) performance.

Ultimately, the SPIRIT system will be designed to encompass edge computing capabilities and heterogeneous access network technologies, including but not limited to 4G/5G, Wi-Fi, and fixed broadband connections. Furthermore, the system aims to be compatible with a wide array of end-user devices, such as headsets and mobile devices. This flexibility will allow end users to seamlessly access and utilise the applications supported by the SPIRIT system according to their preferences. In addition, the SPIRIT system is located across two distinct sites situated in Berlin (Germany) and Surrey (UK). The two test sites are interconnected to facilitate collaborative immersive telepresence applications on a Pan-European scale.

The second version of the innovations enhances the platform with new functionalities and aligns with the SPIRIT evolved architecture to support multi-user scenarios, in particular one-to-many conferencing use-cases.

1.1 PURPOSE OF THIS DOCUMENT

Deliverable 3.2 – "Innovation Platform Enablers (Second Version)" is the second version of a document series and will cover the innovation platform enablers considering use case requirements and system architecture specifications documented in D2.2 [2]. The documentation on the components and implementation of the second version of the SPIRIT platform can be found in D4.2 [5]. The focus of D3.2 is on extending preliminary innovations in the first report to support immersive telepresence applications. These telepresence applications are based on the application platforms implemented by the project partners as part of the SPIRIT project.

1.2 STRUCTURE OF THIS DOCUMENT

The document is structured as follows: in Section 2 we provide a mapping of innovations to SPIRIT architecture and present implemented solutions on immersive telepresence. Afterwards, in Section 3, we describe the innovations that have been developed and are aimed to extend the immersive telepresence solutions. A conclusion for the work done for WP3 is drawn in Section 4.

2 IMMERSIVE TELEPRESENCE SOLUTIONS

Immersive telepresence refers to an advanced communication technology that creates a highly realistic environment that allows users to interact remotely as if they were physically present in the same location [6] [7]. Immersive applications can be used in a broad range of use-cases that extend the existing 2D video conferencing into more authentic real-time 3D services realised on todays' 5G networks. For this project, the innovations developed include consumer-grade devices that enable the realisation of immersive telepresence, making immersive experiences available to a wider consumer community without incurring additional infrastructure costs.

Utilising innovative audio-visual technologies, immersive telepresence goes beyond traditional video conferencing by replicating the nuances of digital human-to-human interactions, including spatial awareness and realistic visuals. Display technologies, such as augmented reality (AR) or Mixed Reality (MR), further enhance the immersive experience. Overall, applications of immersive telepresence span a variety of fields and offer transformative opportunities for collaboration in business, healthcare consultations, education, and other areas.

As the demand for more effective remote communication continues to grow, immersive telepresence holds the promise of redefining how we connect across distances. However, challenges such as bandwidth and real-time requirements as well as cost and interoperability issues must be addressed for widespread adoption.

2.1 EVOLVED ARCHITECTURE AND INNOVATIONS

The innovations are aligned with the evolved architecture in D2.2 [2]. In this section, the mapping of different innovations and extensions to support the evolved architecture are summarized.
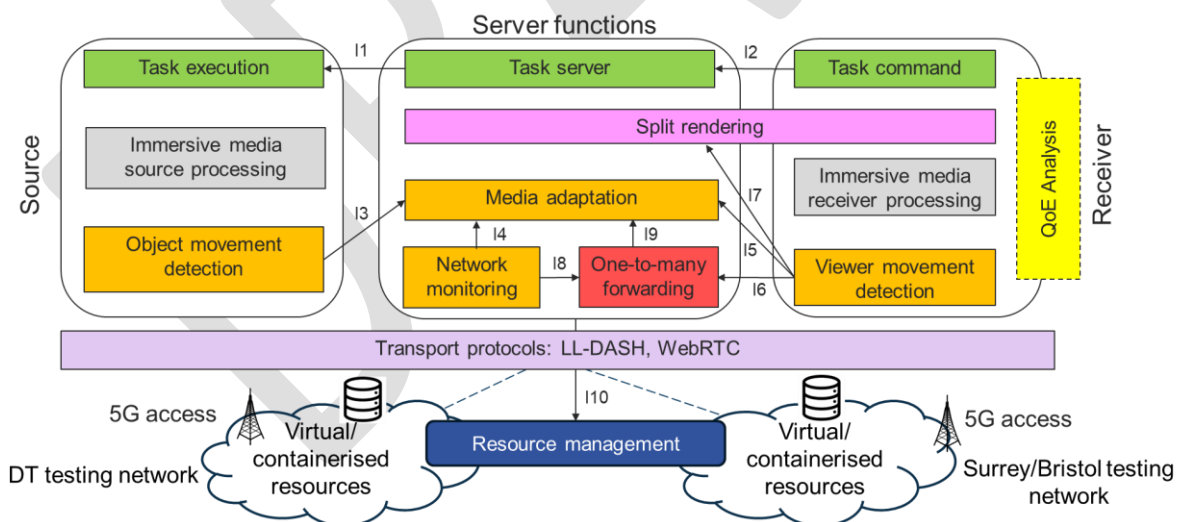
FIGURE 1 SPIRIT PLATFORM ARCHITECTURE [2]

In D2.2, the evolved SPIRIT architecture is described and illustrated in Figure 1 [2] . In general, the SPIRIT innovations consider two different types of models, holographic-based (Section 2.2.1) and avatar-based (Section 2.2.2). In the second version of the platform innovations (D3.2) a new method for video-based avatar generation is described. Although the innovations are primarily described for human-to-human communications, several of the innovations (e.g., transport, security) can be additionally applicable to human-to-machine use-cases.

Two transport protocols are considered in the platform innovations, WebRTC-based (Section 2.3.1) and low latency DASH (Section 2.3.2), and detailed implementations are described in the innovations. In particular, low latency investigations are provided in D3.2.

In addition, immersive media components at the source and the receiver are described for the capturing, processing, and rendering of both holograms and avatars in Section 2.4.1 and Section 2.4.2, respectively. Additional considerations for WebRTC-based media processing are described in D3.2.

Individual innovations that expand the platform capabilities beyond state-of-the-art are considered in the evolved SPIRIT architecture and summarised as follows:

Resource management (Section 3.1): ensures proper allocation of computational and networking resources to support efficient deployment of immersive applications. 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 has been developed.

Resource management to support media adaptation in dynamic is also considered. Specifically, multi-dimensional media adaptation and frame synchronization are studied in D3.2 that take into account the network resource availability, the adaptation decision-making also considers the context information from both the receiver and the source side are detailed.

Scalability (Section 3.2): additional innovations to support adaptive one-to-many conferencing based on multi-description coding (MDC), where all points belonging to a captured point cloud frame are divided in distinct subsets or descriptions, and on a central selective forwarding unit (SFU), where the appropriate descriptions are forwarded to multiple clients. In D3.2, the extensions to support many-to-many use-cases are additionally studied.

Split rendering (Section 3.3): 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. The extension of split rendering to support one-to-many communications is considered in D3.2.

Security (Section 3.4): a survey of security innovations for software deployments in cloud centers as well as intrusion detection mechanisms and their applications for SPIRIT testbed are described in D3.2. The security mechanisms are end-to-end and include the source, server, and receiver functions and can be applicable for different use-cases.

2.2 CONTENT

To facilitate the realisation of true and engaging telepresence experiences, the focus is on the real-time capture of immersive data from end-user devices such as three-dimensional (3D) cameras, which are widely available on the market today. In this context, the term 'immersive telepresence content' encapsulates an approach to construct 3D representations of people authentically reflective of the participants.

The endeavor to achieve true immersion in telepresence applications involves a comprehensive technical strategy centered on content creation. Within this project, this strategy recognises the crucial significance of photorealistic avatars and 3D representations of human participants, hereafter referred to as holograms, in enhancing the overall telepresence experience. The beginning of WP3 is dedicated to supporting the intricate process of generating and rendering

such content, using innovative techniques to achieve a seamless merging of realism and interactivity.

The following solutions on immersive telepresence content have been implemented and integrated by the partners into the SPIRT platform testbeds.

2.2.1 Holograms

Prior to exploring technical considerations of generating 3D representations, e.g., human holograms, it is crucial to underline the profound significance of capturing authentic experiences.

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

In our set up the content for immersive telepresence use-cases includes 3D data representing a human face/torso. The 3D data originates from RGB and depth information obtained in real-time using a commercially available shelf depth camera. The camera is equipped with a Software Development Kit (SDK) that offers Application Programming Interfaces (APIs). Utilising these APIs, the RGB and depth frames are synchronised and processed to generate a point cloud.

The point cloud data is subsequently transformed into meshes. Meshes in computer graphics are digital 3D structures composed of interconnected vertices, edges, and faces. Vertices are points in space, edges connect these points, and faces represent the surface between the edges, forming polygons. The preference for meshes over point clouds primarily stems from their advantages in visualisation. Meshes offer a more visually intuitive representation of 3D data compared to point clouds, enhancing human interpretation, and understanding of the shape and structure of the object. Additionally, the use of meshes aligns with interoperability requirements of this project, as meshes are widely supported in diverse 3D graphics software and game engines [8] [9]. This compatibility facilitates the seamless sharing and integration of 3D models across various applications.

Figure 2 shows a point cloud and Figure 3 shows a corresponding mesh. As it can be seen when comparing the figures, the mesh object visually looks more appealing, since a smoother texture is generated compared to the point cloud.

FIGURE 2: POINT CLOUD EXAMPLE
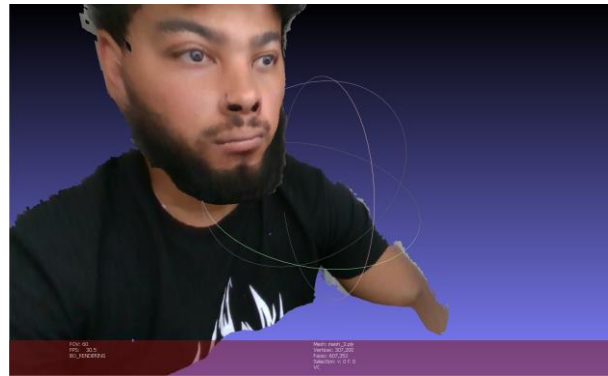
FIGURE 3: MESH EXAMPLE

2.2.2 Photorealistic Avatars

Avatars that represent humans with a high degree of realism can be generated by using the Skinned Multi-Person Linear Model SMPL as starting point. To create a personalised 3D avatar, an actor is captured in the volumetric studio of the Fraunhofer Heinrich Hertz Institute (HHI) to obtain 16 stereo-multi-view videos together with mesh-based 3D geometry streams of the captured person (see Figure 4Figure 4). As shown in the image, lights are installed in the floor, the ceiling, and the walls to create a diffuse light situation. Stereo video cameras are mounted on the walls.

FIGURE 4: IMAGES OF THE VOLUMETRIC CAPTURE STUDIO IN HHI

Based on the captured data a parametric representation of the body movements is extracted, as well as the facial expressions.

The software developed for body model fitting automatically retrieves all the required data from the volumetric capture data server. It then converts and rotates the input images to perform skeleton extraction from the original 2D camera data. The studio camera calibrations are used to align the skeleton estimations from the camera views into 3D. In a next step, the volumetric mesh surface is used as an optimisation target to find the SMPL shape and pose data. The result of processing a volumetric take is a data file that contains shape parameters of the model in the take, and a set of joint angles for each volumetric frame. Processing is supported for SMPL, SMPL+H and SMPL-X. The shape and pose data file can then be used in subsequent tasks of model generation.

The tracking of facial expression is performed with a personalised blend-shape geometry model using automatically detected facial landmarks and optical flow. In order to increase the accuracy of captured facial expressions, dynamic face textures are extracted additionally (i.e. one face texture for each captured frame) from the multi-view footage using a graph-cut based approach [10]. While the hybrid representation (i.e., dynamic texture + dynamic geometry) improves and simplifies the facial performance capture process, it also makes the animation of the created 3D avatar more difficult as facial expressions cannot be represented by a single low dimensional parameter vector. To circumvent this limitation, a deep generative face model based on a variational auto-encoder architecture is trained. During training, this auto-encoder learns to reconstruct face geometry together with the corresponding face texture from a low dimensional latent expression vector. This latent expression vector acts as an interface to control the facial expression of the avatar and simplifies the training and integration of different facial animation methods (audio/video based), which are described in the remainder of this section. A detailed description of the facial performance capture procedure and the model creation method (Figure 5) can be found in [11].
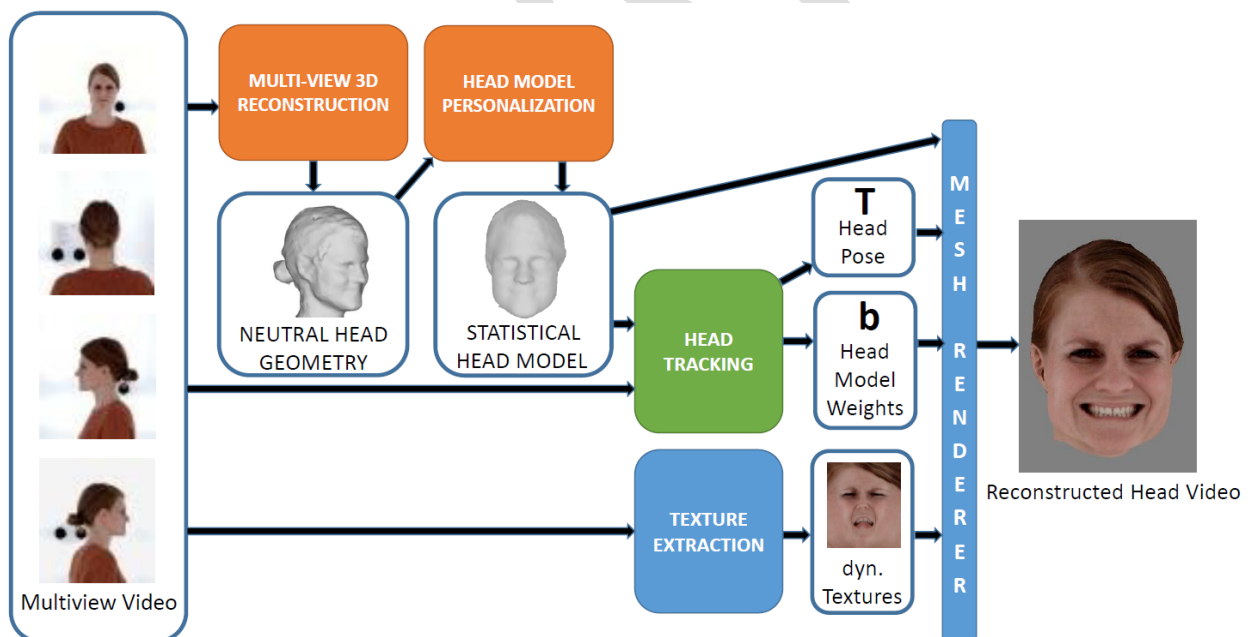
FIGURE 5 ILLUSTRATION OF THE HYBRID FACE CAPTURE APPROACH

2.2.2.1 Audio-Driven Animation

The animation approach integrated in the demonstrator consists of two parts: an audio-based component that performs face/head animation if speech is detected and a rule-base component that performs example-based idle-animation of head and face if no speech is detected. This rule-based component synthesises idle body animations as well.

Since the avatar is created from a single person dataset (with captured speech audio) training a multi-person speech-based animation model is difficult. To circumvent this limitation, a pre-trained speech-feature network [12] that transforms raw 16 kHz audio samples into subject-independent speech feature vectors (at 50 Hz) is employed. Each speech feature holds the un-normalised phoneme class likelihood. To animate the avatar's face, the last 32 speech features vectors are processed with a Convolutional Neural Network (CNN) [13] transforming them into the current animation parameter for the 3D avatar. To differentiate between active speaking and noise, on the predicted class likelihoods are used, where the first entry of each feature vectors indicates no-speech. To avoid unnatural facial animations caused by non-speech noise, audio-based facial animation is disabled after approx. 0.6 seconds without detected speech and use an example-based animation method as fall back. The example-based animation approach randomly selects idle face sequences from a manually created animation database. These idle sequences are then simply re-played by the avatar. Further exponential filtering is employed to minimise temporal artefacts at the transition points between concatenated idle sequences. The random playback of idle-face sequences continues until speech is detected again.

To avoid a stiff and unnatural body posture, a simple rule- and example-based animation approach is used for the body. Similar as for the face, an animation database of idle body sequences is created. This database contains three types of idle data: sequences with very little motion, sequences with stronger motion as well as sequences with very distinct movements (e.g., arms swing, rotation of the torso, etc.). During speech, only random idle sequences with little motion are selected to avoid a mismatch between spoken content and body language. If the avatar does not speak, sequences from all three classes are selected randomly: little motion with a probability of 50%, medium motion with a probability of 30% and distinctive motion samples with a probability of 20%. Again, exponential filtering is used to reduce temporal artefacts at the transition between concatenated samples.

2.2.2.2 Video-Driven Animation

An interesting avenue is extending the audio-driven animation towards video-driven animation. This extension would allow video-based animation of the 3D neural head models derived directly from video footage leveraging the combination of personalized 3D head models and an LSTM-based network. The core idea is to extract universal animation parameters from video-based facial expressions, which enable the creation of lifelike and detailed animations. These animations can be seamlessly integrated into various virtual and augmented reality environments, thereby enhancing user experience in immersive settings.
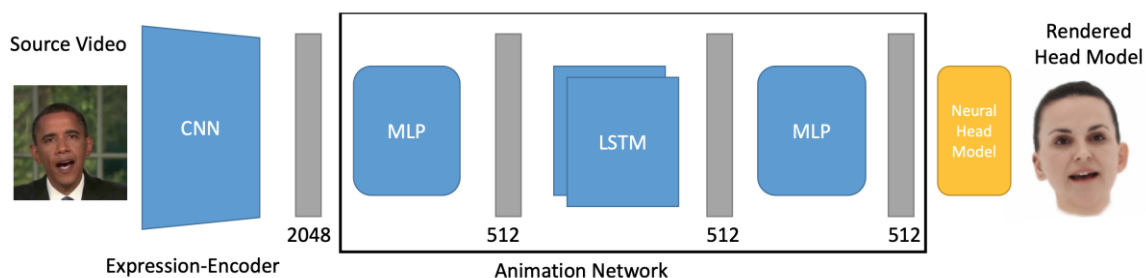
FIGURE 6 PROPOSED ARCHITECTURE OF VIDEO-DRIVEN ANIMATION

A significant technical challenge addressed by this approach is the limitation of traditional animation methods, which often fail to capture both the geometric intricacies and photorealistic textures required for true-to-life facial representations. The approach will utilize the hybrid models and advanced neural rendering techniques that are already part of the audio-based animation.

These techniques ensure the preservation of fine geometric details and texture fidelity, which are critical for accurately representing the subtle nuances of facial movements and expressions.

The pipeline for animating 3D neural head models from video inputs involves several key stages. First, a high-quality 3D head model is generated for the subject based on captured video data. Then, relevant features are extracted from the source facial video. An LSTM (Long Short-Term Memory) neural network processes the captured expressions to derive universal animation parameters, enabling the system to animate the neural head model accurately based on the expressions. An overview of this proposed architecture is shown in Figure 6.

In addition to developing these core technologies, we will address the integration of the method into existing 3D rendering pipelines. This integration is key to ensuring that the developed technology can be easily applied across different platforms and applications, without requiring significant modifications to current systems. The goal is to provide a versatile solution that can be widely adopted in various technological contexts, from entertainment and gaming to professional training and remote communication.

2.3 TRANSPORT

The SPIRIT platform is intended to be a scalable platform for innovation in the field of immersive real-time telepresence. Therefore, the underlying transport mechanisms for data streaming should offer scalability and low latency.

The transport of media and data is primarily realised utilising WebRTC[1] (Web Real-Time Communication). Investigations on LL-DASH[2] (Low Latency Dynamic Adaptive Streaming over HTTP) are incorporated in this report.

2.3.1 Web Real-Time Communication (WebRTC)

In general, WebRTC can be seen as a set of protocols, APIs and standards that work together to enable real-time communication over peer-to-peer (P2P) networks. WebRTC is designed for ease of implementation and eliminates the need for additional plugins or complex configurations, expediting the development process. Furthermore, WebRTC is an open-source technology, fostering a vibrant developer community and promoting innovations.

A collection of protocols is standardised by the Real-Time Communication in WEB-browsers Working Group at [14] of the Internet Engineering Task Force (IETF) while new sets of APIs are standardised by the Web Real-Time Communications Working Group [15] of the World Wide Web Consortium (W3C).

A key benefit of WebRTC is the ability to establish direct P2P connections. By minimising the need for intermediate servers, WebRTC reduces end-to-end latency and ensures smooth interactions in real-time applications, such as video conferencing and collaborative tools [16].

Furthermore, WebRTC inherent strong commitment to security and privacy [17]. It offers end-to-end encryption for all communication profiles, protecting sensitive information from potential

---

[1] https://webrtc.org/

[2] https://dashif.org/

eavesdroppers and hackers. This robust security framework makes WebRTC a desirable choice for applications that prioritise user data protection and confidentiality.

In the dynamic landscape of real-time communication technologies, the decision between adopting WebRTC or alternative solutions is shaped by several critical considerations.

- WebSocket [18] excels in facilitating bidirectional, real-time communication with a primary focus on efficient data exchange. However, it falls short in addressing the comprehensive multimedia needs fulfilled by WebRTC. The latter seamlessly integrates audio and video communication within its framework, making it a more suitable choice for applications requiring a robust multimedia experience.

- Session Initiation Protocol (SIP) [19], stands as a widely recognised protocol for voice and video over IP. Despite its prevalence, SIP introduces complexities by often necessitating additional protocols for media transmission. In contrast, WebRTC offers a unified and self-contained solution, streamlining the real-time communication process without the need for supplementary components.

- The H.323 protocol [20], akin to SIP, establishes standards for real-time communication over IP networks. However, its implementation can be intricate, demanding a comprehensive understanding of various components. The simplicity and accessibility of WebRTC makes it a pragmatic choice for developers seeking an efficient solution without compromising functionality.

In summary, the advantages of WebRTC over similar technologies are underscored by its adherence to open standards, ensuring interoperability, and native support in major browsers, offering a consistent user experience across platforms. Its seamless integration of audio, video, and data communication provides a rich multimedia experience suitable for various immersive telepresence application that consider real-time communication as well as security and privacy. One drawback of WebRTC is the need of additional server instance(s) for signalling purposes in case of strict network configurations in the form of Network Address Translators (NATs) and firewalls, as often the case for corporate networks.

2.3.1.1 Architecture

Figure 7Figure 7 shows an overview of the general WebRTC architecture:
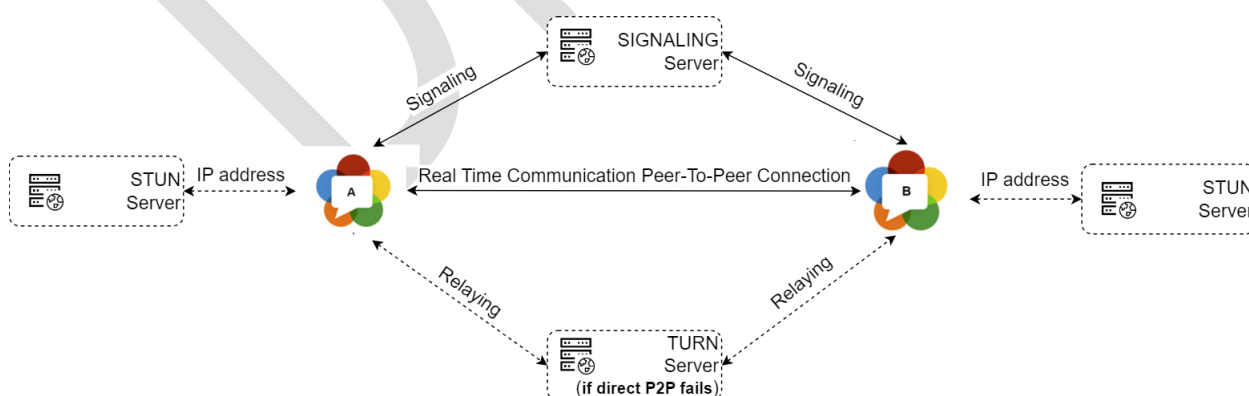
FIGURE 7: WEBRTC ARCHITECTURE

Fundamentally, the architecture consists of two peers that want to communicate with each other, as marked in figure with "A" and "B".

To bridge the gap between these peers, a signalling process is employed. Signalling is the mechanism through which the peers exchange vital metadata, including information about their media capabilities and network addresses. This information exchange is pivotal for the WebRTC connection set up. However, it does not take place via a WebRTC protocol itself, but via external means such as WebSocket, which must be implemented into a signalling server. Accordingly, the signalling server works as an intermediary that ensures that the peers can recognise each other and exchange important metadata before actual data and media streaming is conducted.

To overcome the hurdles presented by NATs and firewalls, WebRTC can leverage STUN Session Traversal Utilities for NAT (STUN) [21] -and Traversal Using Relays around NAT (TURN) [22] servers. STUN servers help in discovering public IP addresses and port mappings, while TURN servers function as relays if a direct P2P connection encounters obstacles due to network restrictions. WebRTC uses protocols such as Interactive Connectivity Establishment (ICE) [23] to collect information about potential STUN and TURN servers.

2.3.1.2 Connection Set Up

The process of setting up a WebRTC connection involves several vital steps. Figure 8 shows an overview of the WebRTC connection set up procedure assuming peer A is behind a firewall with symmetric NAT.
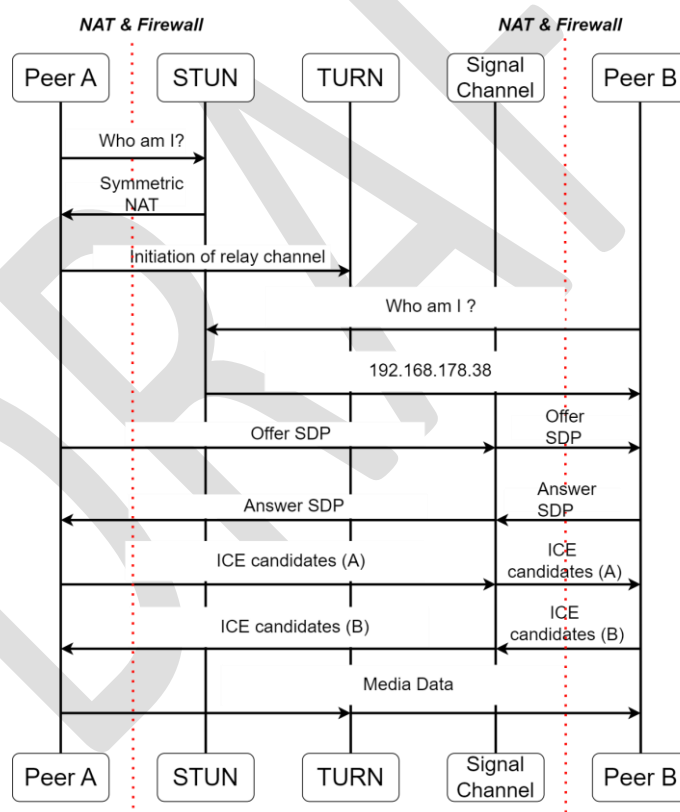
FIGURE 8: WEBRTC CONNECTION SET UP

In the initial phase, peer A, the caller, initiates the communication session with peer B, the callee. To do so, the peers first try to retrieve their public IP address with the help of respective STUN servers. However, since peer A assumes symmetric NAT, as shown in Figure 8, peer A uses a TURN server instead to establish a relay channel. With the relay channel, the peers can bypass the network restrictions originating from symmetric NAT.

In the next step, Session Description Protocol (SDP) [24] offer -and answer messages [25] are exchanged. SDP is a protocol that describes multimedia communication sessions and includes information about media streams and codecs. Peer A starts the SDP exchange by creating an offer SDP. The offer is sent to peer B through a signalling server. Upon receiving the offer, peer B generates a corresponding answer SDP and sends the corresponding answer to peer A, through the signalling server as well. In essence, the SDP negotiation process forms the foundation of WebRTC connection set up, ensuring that both peers are aware of each other's media capabilities.

The ICE protocol is used to gather potential ICE candidates. A candidate includes a combination of IP address and port for a particular transport protocol. Candidate types include those derived from local interfaces (host candidates) like Ethernet or Wi-Fi or obtained through tunnels such as Virtual Private Networks (VPNs). A WebRTC peer uses STUN or TURN to get additional candidates, including server-reflexive candidates from the public side of a NAT and relayed candidates from TURN servers. If using only STUN servers, the peer obtains only server-reflexive candidates. It is important to note that ICE candidates can also be exchanged via SDP messages by gathering them prior to connection set up [26].

Once the SDP messages and ICE candidates have been exchanged and processed, the peers have the necessary information to set up a connection. They utilise the received ICE candidates to create UDP or TCP streams directly between their devices. Depending on the pre-selected communication profile, the P2P link can be used for real-time audio, video, or data streaming, enabling interactive and immersive user experiences.

2.3.2 Low-Latency Dynamic Adaptive Streaming over HTTP (LL-DASH)

Besides WebRTC, which is mainly used in real-time communication scenarios, e.g., video conferencing, HTTP Adaptive Streaming (HAS) is a wide-spread technology for media streaming services, mostly used for on-demand streaming, but also applicable for live streaming. One of the main advantages of HAS over WebRTC is scalability [27]. Today, HAS is available in two prevalent variants: MPEG-Dynamic Adaptive Streaming over HTTP (DASH) [28] and Apple's HTTP Live Streaming (HLS) [29]. In HAS-based systems, media sequences (typically, videos) are split timewise into short- and fixed-duration segments (typically, between one- and ten-seconds durations), and each segment is encoded into multiple versions (quality levels) called representations (typically differing in bitrate and/or spatial resolution). Information on the segments, including their locations on media servers, is stored within a manifest file, in DASH named *Media Presentation Description* (MPD). The video segments and manifest files in HAS are typically delivered through a network of servers and other equipment, referred to as a content delivery network (CDN), that work together to scale video delivery systems and reach end users. HAS players of end users then process MPDs and consider the current network conditions (e.g., effective download throughput) and/or capabilities of the user equipment (e.g., the playout buffer) to adaptively download appropriate representations from media servers using adaptive bitrate algorithms, aiming at achieving high QoE for the users.

The end-to-end latency of HAS-based systems is typically in the order of (many) seconds, due to the complex distributed workflow sketched above (involving, e.g., cloud-based encoding and packaging, MPD generation, distributing content and manifest to/within a CDN). This makes basic DASH- and HLS-based systems infeasible for real-time telepresence systems. However, in recent years, low-latency (LL) versions of DASH and HLS were developed, with the aim to reduce end-to-end latency of live streaming scenarios to a few seconds. Most recently, a LL-DASH version was reported to achieve less than 600 milliseconds (on average) of end-to-end latency in bidirectional live sessions involving point cloud content, "*right at the edge of what is considered acceptable for 2D videos*" [30]. This seems to make live LL-DASH feasible to be used in SPIRIT for immersive telepresence use cases.

LL-DASH makes use of the MPEG Common Media Application Format (CMAF), a standard transport container for streaming, and, originally, of Chunked Transfer Encoding (CTE), a data transfer mechanism in HTTP/1.1. (HTTP/2 has other data transfer mechanisms to be used similarly.) Based on CMAF in low-latency mode and CTE, a media player can request incomplete media segments. A segment then is a collection of one or more smaller media units called *chunks*. Chunks are the smallest referenceable media units and can be of sub-second duration, depending on the encoder configuration. (There is a potential latency–quality trade-off, though.) Figure 9 illustrates live low latency streaming over HTTP using CTE.
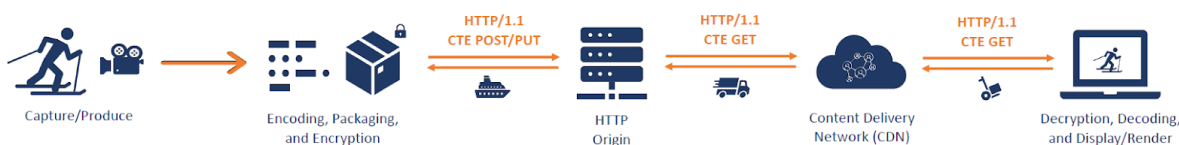

FIGURE 9: SKETCH OF LL STREAMING OVER HTTP USING CTE (AND AN OPTIONAL CDN) [31]

Using CTE, the producer (sender) can deliver chunks of data of undetermined size, which enables transferring the generated CMAF chunks as they come out from the encoder and packager, and without the need to be written on the disk (if caching the data is not desired). However, the usual workflow is to make the content available on the HTTP origin (or, ingestion) server. Media players can request, receive, and decode chunks before the entire segment is complete on the producer side (and the origin server), thus enabling low latency to be achieved. The MPD must be periodically updated by the producer and be repeatedly retrieved by the media players to learn about fresh content. In practice, MPD handling (refreshing and refetching), time synchronisation, and buffering such as to cope with network issues and still ensure smooth playout, make LL-DASH complex to implement and configure.

SPIRIT will consider LL-DASH as a media transport mechanism, as an alternative to WebRTC, due to the following considerations:

- DASH is much more scalable than WebRTC. There are no peer-to-peer connections. Once the content and the MPD are available (refreshed), a significant number of players can retrieve the content from the HTTP origin (or from a content delivery network, if employed) using the simple HTTP mechanisms. It is only the power of the HTTP server(s) that determines how many players can retrieve the content.

- DASH does not have issues with firewalls and NAT devices. DASH works "*over the top* (OTT)", via HTTP, which uses standard (open) ports and is supported on virtually any end device.

- DASH is standardised, vendor-independent, and wide-spread. LL-DASH is not yet too widely used, but there are effective implementations.

- DASH enables adaptive streaming. Multiple quality versions of the content (of segments – and chunks, in the LL case) are produced, allowing the consumers (media players) to retrieve the quality versions fitting their current (network and buffer) conditions.

- DASH hands control over to the media players (rather than having it on the producer side). This enables the players to run their specific media/quality adaptation strategies and supports scalability.

- As indicated above, LL-DASH has been demonstrated to achieve sub-second end-to-end latency in immersive streaming use cases meanwhile [30].

To make LL-DASH available for point cloud streaming and immersive telepresence applications, SPIRIT established contact with the TRANSMIXR project[3], in particular the *Distributed & Interactive Systems Group* at *Centrum Wiskunde & Informatica* (*CWI*)[4] that makes the *VR2Gather* software available open source [30]. The *VR2Gather* software is an outcome of the EC Horizon 2020 VRTogether[5] and the EC Horizon Europe TRANSMIXR projects and supports immersive media applications, among them point cloud content capturing, encoding, transmission (including via LL-DASH), decoding, and rendering. According to the project websites and the publication, quite some components were validated and demonstrated in relevant environments and use cases, thus seemed to be ideal candidates to be adopted by SPIRIT. Double development efforts would be avoided.

At the time of writing of D3.1 (i.e., end of 2023), the *VR2Gather software* was not fully usable for third parties. In particular, the LL-DASH components were still mostly proprietary; open-source versions were under construction. Moreover, the software structure was undergoing a major overhaul and the documentation was being modified and extended.

As a consequence of that status, the *VR2Gather* software was not yet used in the first version of the SPIRIT platform.

As of the time of writing of D3.2 (September 2024), the LL-DASH components of *VR2Gather* are still not fully usable by third parties. *VR2Gather* uses the proprietary DASH transport modules from *Motion Spell*[6], and they are incorporated twofold [30]:

- the LL-DASH sender uploads DASH fragments and Media Presentation Descriptions (MPDs) to a media server.

- the LL-DASH receiver selects the appropriate streaming quality based on the device conditions and obtains the corresponding video tiles.

Since using LL-DASH with *VR2Gather* was not possible for the SPIRIT partners, it was instead decided to expand the capabilities of the *VR2Gather* platform to encompass AR/XR devices, with a particular focus on the *Meta Quest 3*. This would allow SPIRIT to have an AR platform with LL-DASH transmission of point clouds once this solution was made available. (A project successful in the SPIRIT Open Call 1 (Open-DASH-PC) will provide the *VR2Gather* software fully open source and demonstrate its features and performance.)

The integration of AR in *VR2Gather* on the *Meta Quest 3* leverages the headset's cutting-edge pass-through technology, allowing users to seamlessly blend virtual and real-world elements within a shared environment.

To achieve this, the partner responsible for this work (UNI-KLU) implemented a variety of solutions. AR functionalities like plane detection and hand input are now seamlessly integrated, enabling users to place virtual objects from the *VR2Gather* platform onto detected surfaces such as tables and floors. Users can switch between traditional controllers and hand gestures for interaction, as detected by the headset's cameras. This intuitive interaction allows for a more

---

[3] https://transmixr.eu/

[4] https://www.dis.cwi.nl/

[5] https://vrtogether.eu/

[6] https://www.motionspell.com/

natural and engaging experience and the input flexibility empowers users to choose the method that best suits their preferences and the specific task at hand.

Furthermore, the platform's scenes have been remodeled in accordance with the Unity engine's ARFoundation and Meta OpenXR feature packages. This includes the removal of walls and floors, adjustments to camera properties to accommodate pass-through, and a series of modifications added to ensure a more fluid and immersive AR experience.

2.4 APPLICATION

Different platforms to support immersive telepresence applications were implemented within the SPIRIT project. The applications supported by the platforms of the are based on use-case scenarios in which two peers can take part in conversations set in AR/MR environments embedded in 5G network testbeds.

The solutions on immersive telepresence application platforms by Ericsson (EDD) and Fraunhofer HHI presented in the following sections were successfully tested and integrated into the network testbed of Deutsche Telekom (DT) to allow project partners and Open Call participants to develop, implement or test new innovations within one controlled test environment.

In addition, the 5G Innovation Centre (5GIC) provides a fully implemented immersive telepresence application use case integrated in the 5G network testbed hosted at University of Surrey (UoS) that was described in D2.1 [4]. The DT network testbed and the 5GIC UoS network testbed are interconnected allowing the development and test of telepresence applications in scenarios with heterogeneous network conditions.

2.4.1 Real-Time Holographic Communications

Figure 10 illustrates the application platform with a particular focus on immersive telepresence applications supporting real-time holographic communications use cases:
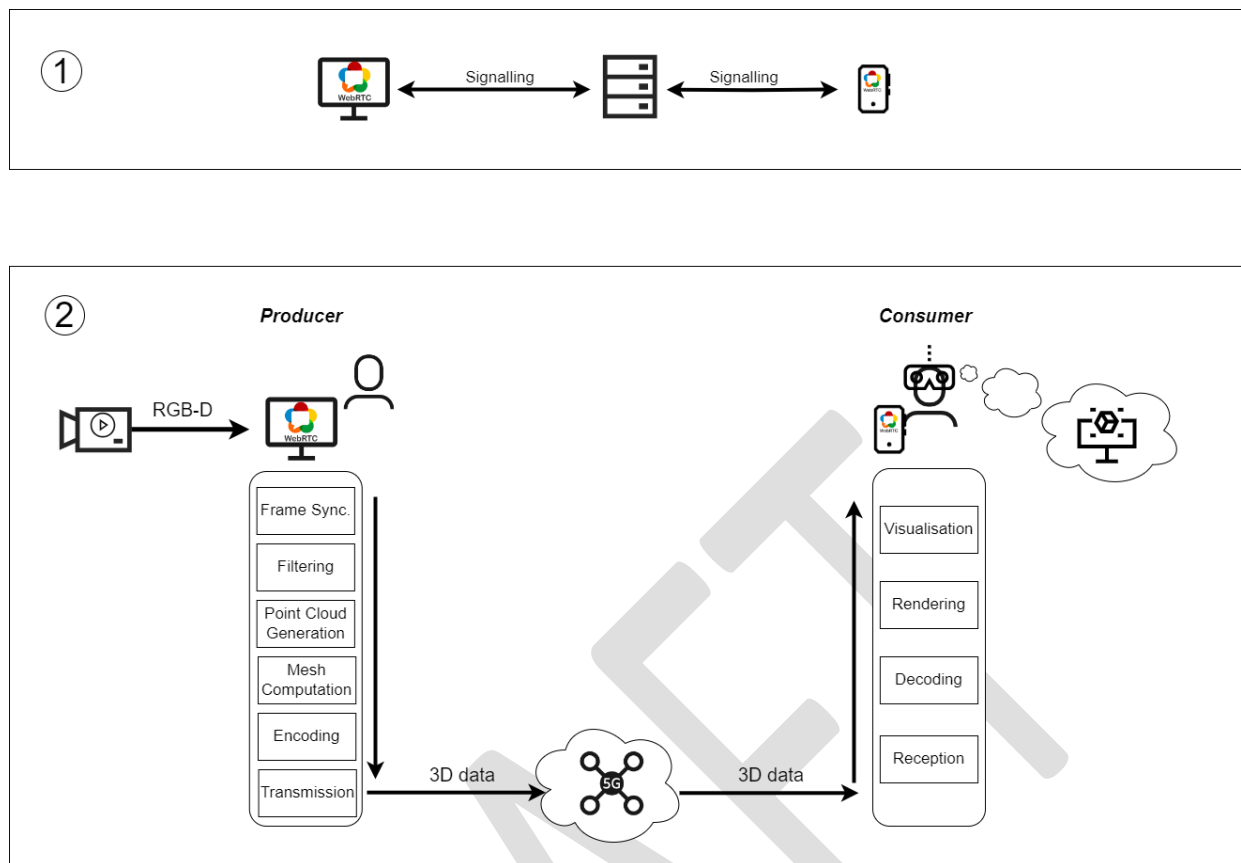
FIGURE 10: HOLOGRAPHIC COMMUNICATION

The application platform performs two primary tasks to provide immersive telepresence. Firstly, the applications on producer and consumer side handle connection set up and data streaming using WebRTC, as detailed in Section 2.3.1.2. The WebRTC P2P data streaming link enables real-time communication commonly associated with digital human-to-human interaction use cases. The second task involves the generation and processing of 3D content in the form of meshes, as outlined in Section 2.2.2. The content delivery is designed to align with QoE demands specified by individual immersive telepresence applications. This necessitates an inherent flexibility, allowing for seamless adjustments, including modification of content, to guarantee an optimal user experience.

In this context, the Python[7] application on the producer side is tasked with capturing, processing, and transmitting 3D data in the form of meshes that represent human features, including those related to the torso and face. Conversely, the Unity[8] application on the consumer side functions as the recipient, responsible for decoding and accurately visualising the received 3D data in real-time. The data stream is facilitated through established wireless or wired network access technologies, such as 5G.

The producer and consumer application apply various processing steps on the data acquired by the depth camera. These steps include techniques aimed at reducing the bandwidth requirements

---

[7] https://www.python.org/

[8] https://www.unity.com/

Funded by the Horizon Europe Framework Programme of the European Union

associated with the complex nature of the 3D data while maintaining satisfactory quality of the human 3D representation at the receiving end.

The producer application consists of the instances that follow a producer-consumer scheme. One challenge in interconnecting those parts arises from the fact that they are a mix of synchronous (encoding) and asynchronous (WebRTC) operations. To overcome this challenge a multi-process buffer is employed offering synchronous and asynchronous buffering.

To cope with higher QoE settings related to resolution and frame rate, the platform offers the ability to make use of multi-processing at the producer side. This approach especially benefits from multiple cores within the hardware the producer is deployed. To prevent rubber banding (i.e., incorrect order of reception of 3D data) at the receiver end due to multi-processing. The application platform employs the mechanism to guarantee multi-process sequence ordering.

When it comes to WebRTC, the data channel API is used to for streaming the encoded data. Between two peers, it is possible to establish up to 65,534 data channels. Each data channel operates on datagram-based communication and is configurable with its own durability settings. By default, each data channel ensures ordered delivery of transmitted data.

The significant advantage offered by data channels lies in their configurability to emulate the characteristics of UDP (User Datagram Protocol) by enabling unordered and potentially lossy delivery. This feature becomes crucial in scenarios that demand low latency and high performance. It becomes possible to monitor and assess the backpressure exerted and ensure that data transmission aligns with the network's capacity, thereby optimizing the overall performance.

WebRTC utilizes the Stream Control Transmission Protocol (SCTP) as an application layer protocol over the DTLS connection. SCTP offers configurable streams, which are encapsulated by WebRTC data channels. To address features not expressible by SCTP, WebRTC employs the Data Channel Establishment Protocol (DCEP) to communicate channel labels and protocols.

The use of SCTP for the WebRTC data channel communication offers key features such as multiplexing, reliable delivery via a TCP-like retransmission mechanism, partial-reliability options, congestion avoidance, and flow control.

The adaptability of reliability and latency configurations in the network context justifies the use of WebRTC data channels in the holographic communication application platform. Consequently, the platform provides users with greater flexibility to meet specific Quality of Experience (QoE) requirements pertaining to network aspects.

2.4.2 Real-Time Animation and Streaming of Realistic Avatars

The Real-Time Animation and Streaming of Realistic Avatars use case proposes a scenario of asymmetric communication between two users, one acting as generator of a certain type of media (audio, text) and the other as receiver of both video and audio signals. A pre-captured avatar (Section 2.2.2) is animated from the generated media on an edge cloud server, providing updated mesh and texture in real-time. To overcome the bandwidth requirements of the continuous transmission of this kind of complex 3D objects, the avatar is rendered on the server frame by frame, providing a 2D image that is encoded and sent, together with an audio signal, to the receiver user. To do so, WebRTC is used as streaming mechanism, achieving the required performance in terms of latency and security. To ensure the proper perception of the object as a 3D element, the viewpoint of the receiver user is continuously synchronised on both client and server applications. The general architecture of the use case can be seen in Figure 11.

The system is based on the concept of Split Rendering. On the server, the first part of the rendering is performed. At this stage, the image provided by a virtual camera, consisting of a specific view of the avatar with a monochromatic background is generated, encoded, and transmitted. Then, the second part of the rendering process takes place on the receiver client application. Here, the background is detected and removed, integrating the avatar in the video signal provided by the real camera of the device. This improves the immersion of the user since the avatar is integrated in the scene.
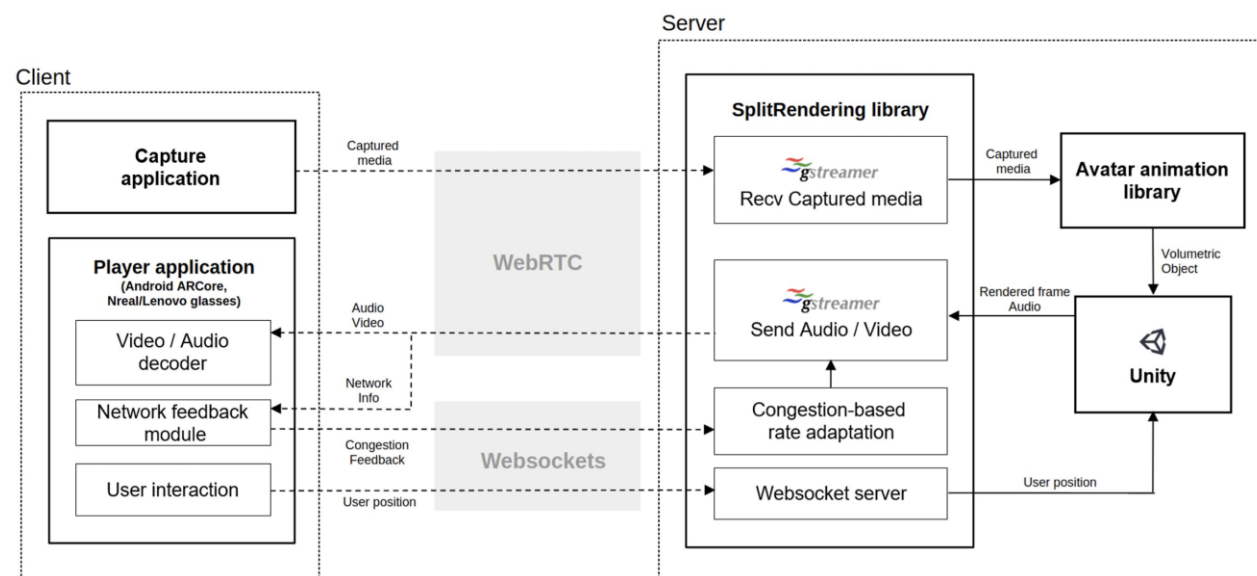

FIGURE 11: ARCHITECTURE OF PLATFORM FOR REAL-TIME ANIMATION AND STREAMING OF REALISTIC AVATARS

As mentioned, there are three main elements in the system:

1) **Client capture application:** native Android app that captures media (audio, video, text) and transmits it through the network using a WebRTC connection.

2) **Server animation, rendering and streaming application:** Unity app running in a Linux environment on a cloud edge server. It performs the following steps to generate the rendered video and audio.
   a) Connections from both the sending and receiving clients are established via WebSocket.
   b) The media captured by the client application is received at the server through a WebRTC connection.
   c) This media is decoded and processed on a Gstreamer pipeline and then used as input to the avatar animation library.
   d) The avatar animation library uses a neural network trained with data recorded from a real person to generate a dynamic 3D mesh and a texture that are recreated as an object in Unity and updated frame by frame. The current viewpoint of the receiver user is considered to animate the avatar, i.e., to direct its gaze.
   e) Once the mesh and texture of the avatar for a frame is obtained, the viewpoint of the user is again used to place one or two cameras (in single and stereoscopic modes) in the scene accordingly. The view of this cameras will be rendered and a 2D image of the desired resolution will be generated. In the stereoscopic mode, this image will be the combination of the result of the rendering from both cameras, side by side.

f) The 2D image is inserted, together with the audio that corresponds to the current frame, in the beginning of a GStreamer pipeline. Both media tracks (audio and video) are encoded and sent over the network via WebRTC. This Gstreamer pipeline will be adapted by following the network conditions, according to the information provided by a congestion control algorithm (Low Loss and Scalable Throughput (L4S)).

The architecture of the server application is illustrated in Figure 12Figure 12:
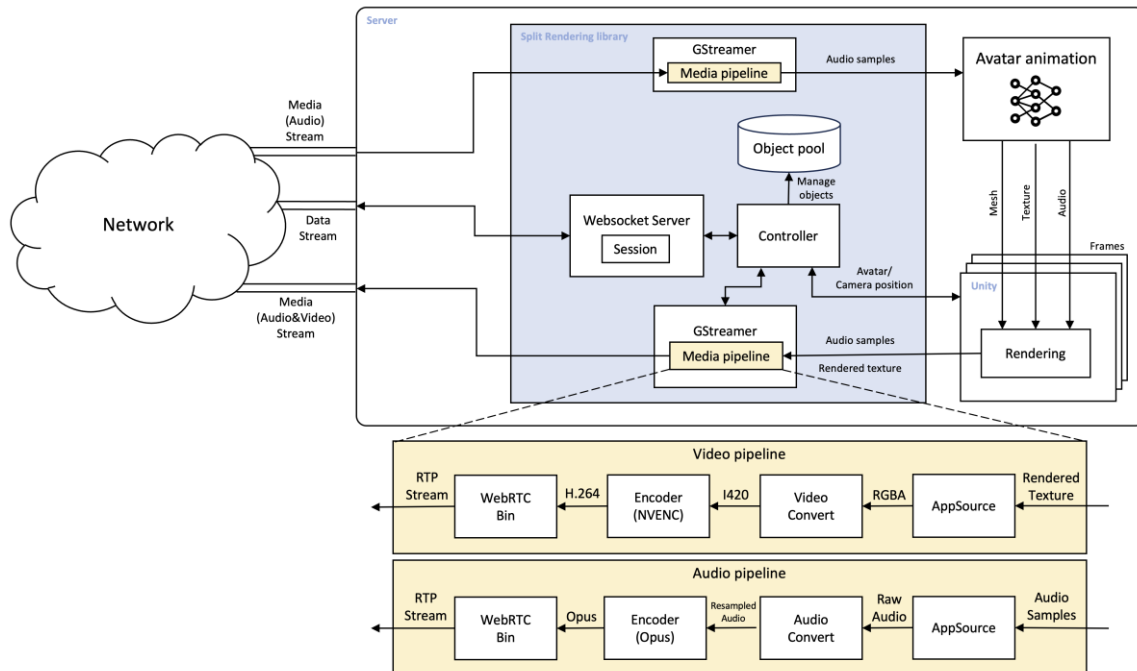
FIGURE 12: ARCHITECTURE OF THE SERVER APPLICATION OF THE PLATFORM FOR REAL-TIME ANIMATION AND STREAMING OF REALISTIC AVATARS

3) **Player application:** Android Unity app that receives the rendered video and audio from the server via a WebRTC connection. It performs the final stage of the rendering by removing the monochromatic background that surrounds the avatar and integrating it in the real scene. It leverages the WebRTC data channel to send feedback to the server. This information includes data such as the viewpoint of the user in the scene, the position and rotation of the avatar and information about the current state of the network, based on a congestion control algorithm like L4S. The receiving user can use two types of devices:

a) Android tablet/phone: the avatar will be shown directly on the screen in a MR environment. The position and rotation of the avatar, as well as its scale, can be modified through the UI of the application.

b) Mixed reality glasses (Xreal Light, Lenovo ThinkReality A3): the client application will run on an Android phone connected to the glasses. The position of the avatar can be modified via a drag 'n drop mechanism. In this mode, a combination of two 2D images is sent, corresponding to the stereoscopic mode.

In all cases, a head rotation feature has been implemented, so the avatar will direct its gaze towards the user.

2.5 SUMMARY

Table 1 provides a summary of the implemented application platforms presented in the previous section.

TABLE 1 : APPLICATION SPECIFICATIONS

	Real-Time Holographic Communications	**Real-Time Animation and Streaming of Realistic Avatars**
Users	Media generator: a person whose 3D representation is computed in the form of meshes and sent to the recipient. Receiver: a user that will visualise the human mesh and interact with it.	Media generator: a person that will generate audio/text to be sent to the server. Receiver: a user that will visualise the avatar and interact with it.
Communication	RTC P2P (WebRTC)	RTC P2P (WebRTC)
Network Access	Wireless, e.g., Wi-Fi / 5G Wired, e.g., Ethernet	Wireless, e.g., Wi-Fi / 5G Wired, e.g., Ethernet
Hardware Components	Depth Camera (Intel RealSense): acquisition of media and spatial information. High-performance PC (Windows/Linux): computation of meshes and data transmission. Mobile phone (Android): decoding and rendering of received data. AR glasses (Xreal Light): visualisation of 3D representation to user.	Edge cloud server: main rendering element in charge of generating the mesh and texture for each frame, render the avatar and send audio and video to the receiver user. Mobile phone/tablet (Android): mobile device with a player application that will receive and show audio and video of the avatar, while allowing interactivity by the user. Mixed reality glasses (Xreal Light, Lenovo ThinkReality A3, Meta Quest 3): mixed reality device either standalone or connected to a phone that will receive and show audio and video of the avatar, as well as allow interactivity by the user.

In the original configuration, both platforms incorporate two users engaged in digital interactions. One of the users, referred to as the media generator provides data to be processed and transmitted to a receiver for the presentation of volumetric content, manifested as a human hologram or photorealistic avatar visible on AR glasses in real-time. The application platforms were integrated into the 5G network testbed provided by DT and UoS. A description of the integration efforts for both application platforms can be found in D4.2 [5].

The updated version of these use cases describes an enhanced video-driven avatar animation to improve user representation in the conferencing use-cases and additional investigations media transport using WebRTC and low-latency DASH protocols.

3   INNOVATIONS

This section presents additional innovations aimed at overcoming the challenges of immersive telepresence. The innovation are set to extend the application platforms described in Section 2, which were implemented based on use case requirements and architecture specification in D2.1 [4] and evolved architecture in D2.2 [2]

Our examination delves into innovative concepts in resource management such as network-based orchestration for platforms like Kubernetes[9] and user-driven network customization.

Additionally, we explore innovative video technologies, including WebRTC and LL-DASH. More specifically, we aim to improve the scalability of the system to support not only one-on-one communication, as currently present in the implemented application platforms, but also one-to-many or many-to-many conferencing applications.

A significant focus of our efforts involves driving innovation in real-time interactions incorporating rendering through a pioneering technique known as "Split Rendering" or "Remote Rendering". This approach aims to shift GPU-intensive software operations from the consumer graded device to a cloud environment, harnessing the computational power of sophisticated hardware. This rendering approach not only optimises end device battery usage but also contributes to reducing end-to-end latency in the overall system due to faster processing time on the software side.

The importance of the innovations presented in the following lies not only in their standalone capabilities, but also in their potential integration with the implemented application platforms. By linking novel approaches with the platforms of the project partners, we aim to improve the telepresence experience across various levels for the SPIRIT platform.

The innovation platform enablers were integrated into testbeds as described in D4.2 [5] and ongoing investigations are ongoing for the integration of new innovations.

3.1   RESOURCE MANAGEMENT FOR TELEPRESENCE APPLICATIONS

Resource management is a crucial undertaking within the SPIRIT platform, given that bandwidth, computational power, memory, and other resources associated with immersive telepresence applications are finite and, consequently, can be costly. The effective handling and provisioning of resources plays a vital role in sustaining QoE demands inherent in immersive telepresence use cases. Simultaneously, efficient resource management ensures the scalability of the platform, allowing to adapt and meet the evolving demands of users and applications.

3.1.1   Network-Aware Resource Scheduler

To avoid monolithic architectures, telepresence solutions typically consist of multiple individual microservices or components, in the form of containers, that together represent a service function chain. To ensure that the deployments of such chains still meet the latency and throughput requirements of the application, resource management and orchestration platforms should be made aware of the network characteristics as well as of computational resource usage. Typically,

---

[9] https://kubernetes.io/

Kubernetes (K8s) deployments do not take networking characteristics into account and primarily focus on central processing unit (CPU) and random-access memory (RAM) optimisation.

This section describes Diktyo, a network-aware scheduling framework for Kubernetes that supports both computational and network resource efficiency.
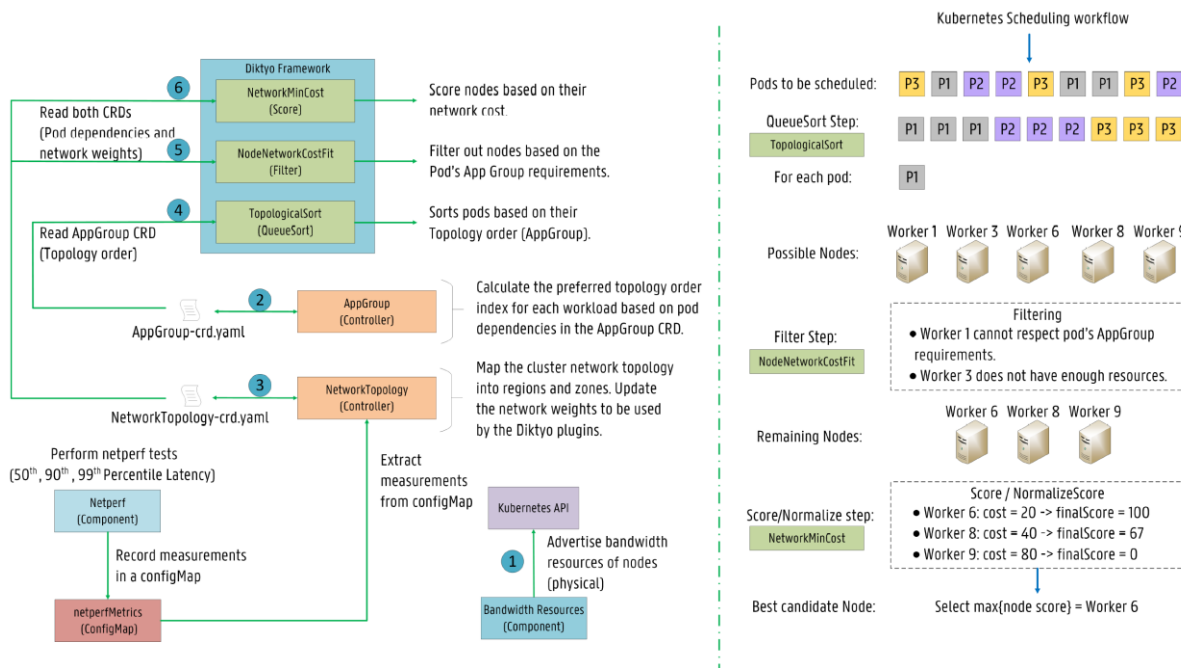

FIGURE 13: DIKTYO FRAMEWORK AND KUBERNETES SCHEDULING WORKFLOW

Figure 13Figure 13 depicts an overview of the Diktyo framework and the typical K8s scheduling workflow, including Diktyo scheduling plugins. Bandwidth resources are advertised to the K8s API to consider the node available bandwidth in the scheduling process (1). The framework introduces two Custom Resources: AppGroup and NetworkTopology to maintain both the application dependency information (2) and the infrastructure network topology (3). Diktyo considers both application dependencies and the cluster network topology when scheduling pods in K8s. The NetworkTopology controller updates network weights between cluster nodes across regions and zones based on a netperf component. Diktyo provides network-aware algorithms implemented as three scheduling plugins based on the K8s scheduler framework: TopologicalSort, NodeNetworkCostFit and Network-MinCost. First, pods are sorted based on their established dependencies (4). Then, nodes are filtered out based on the pod's AppGroup requirements (5), and finally, nodes are scored based on network weights ensuring low network costs between dependent pods (6). Further explanations are available in [32].

The network-aware scheduler has been tested for a variety of benchmark applications. Simulations show that Diktyo can significantly reduce the network latency for various applications across different infrastructure topologies compared to default K8s scheduling plugins. Also, experiments in a K8s cluster with the microservice benchmark applications show that Diktyo can increase database throughput by 22% and reduce application response time by 45% [32].

The repositories have been made available on GitHub:

- https://github.com/diktyo-io.

The network-aware scheduler plugin is also included in the documentation provided at:

- https://github.com/kubernetes-sigs/scheduler-plugins/tree/master/pkg/networkaware.

Integration work of the scheduler has been finished at the DT testbed and at the UoS testbed.

3.1.2 User-Intent Driven Network Adaptation

In holographic streaming, user intent can be expressed through the enhanced interfaces of user devices. For instance, a user can move closer to a holographic object if interested and moves away from this object when not interested. Accordingly, the holographic helm with position tracking can send requests to the content source to adapt the point density to maintain the same visual experience for a moving user. However, the change in video resolution level has a direct impact on the required network bandwidth (e.g., from less than 30 Mbps to more than 200 Mbps). This presents challenges for the network, as it must detect real-time user intentions and transport network conditions (e.g., path delay and bandwidth) to determine whether the current path can meet the requirements of the new user intent. If not, it may need to switch to an optimal path. To address this, first our proposed framework uses an offline interface to negotiate with the application provider. This negotiation aims to categorise user intent into encoding bits. Additionally, an online interface is employed to capture such user intent and implement potential path redirections. As seen in Figure 14Figure 14 there are several key modules collocated at the Software Defined Networking (SDN) controller:
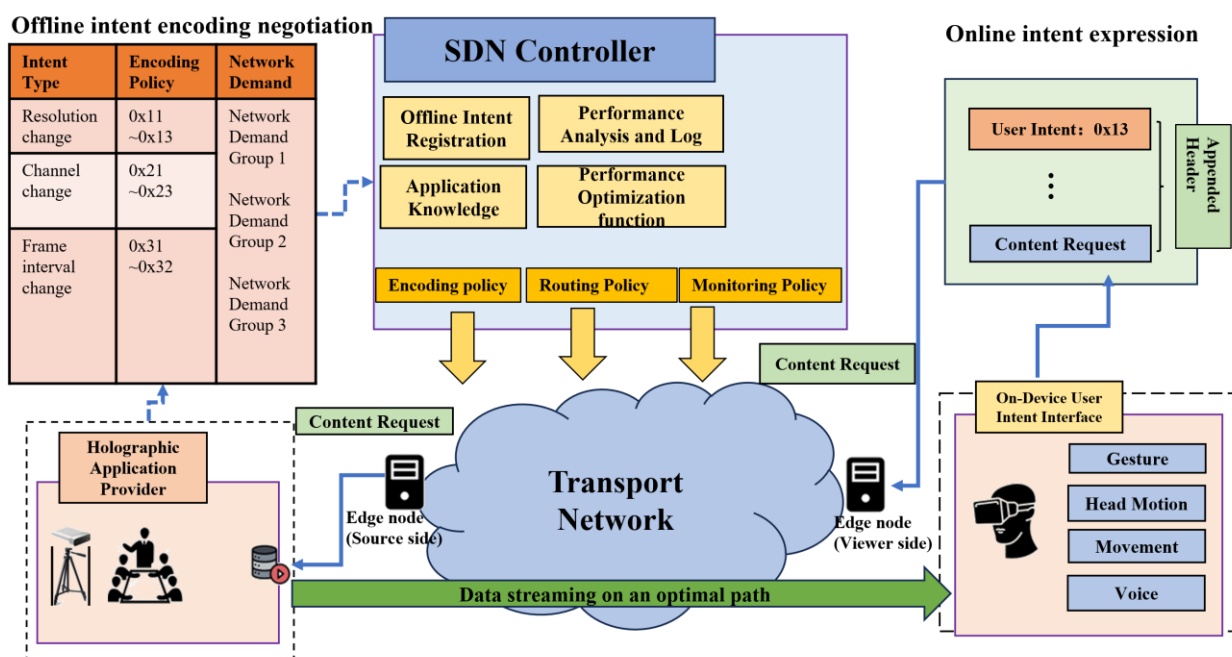
FIGURE 14: USER INTENT DRIVEN NETWORK ADAPTATION FRAMEWORK

1) Offline intent registration module: This module exposes an offline interface for the application provider to register the allowed user intent levels. For instance, user behaviour that changes resolution from lowest resolution level to the highest resolution level can be allocated with intent encoding bits as 0x11, 0x12 and 0x13.
2) Application knowledge module: The SDN controller will translate the registered user intent levels to different network requirements (e.g., bandwidth and delay) according to its own knowledge preloaded for volumetric streaming.
3) Performance analysis and log module: By actively monitoring the volumetric application flows, the SDN controller will save the flow information in this log system with necessary performance inference (e.g., throughput).

4) Performance optimisation module: For different registered user intent types, the SDN controller will generate tailored network adaptation policies to ensure satisfactory performance. For instance, for different resolution level changes, path re-direction can be applied.

By the joint effort of these modules, the SDN controller can generate user intent encoding policy (e.g., how to identify user intent), routing policy (e.g., how to determine optimal path) and monitoring policy (e.g., which flow to monitor) to underlying edge nodes to monitor this specific application flow, identify registered user intent from the appended header of real-time user request. At the same time, the network can maintain a set of optimal paths for each user intent group by probing the path conditions in the online phase. By maintaining an optimal path set for each registered user intent type, the network can decide whether to redirect the incoming user intent to a better path once the current path cannot meet the target requirements.

The repository has been made available on:

- penguos/IntentFramework: Intent-based framework for SPIRIT Project (github.com)

3.1.3 Multi-Dimensional Media Adaptation

In the realm of holographic communications, the realization of highly immersive and interactive experiences is bounded by influence of behavioural factors and limitations in accounting for such factors. This is prevalent in the expression of user intent, with the impact of factors such as user and object behaviour as well as network dynamicity, being significant. Furthermore, when exploring traditional media adaptation mechanisms utilized for typical media applications, there exists limitations in such mechanisms that fail to account for factors more relevant to immersive experiences, thus making such adaptation schemes unsuitable for the realization of user intent and consequently more immersive experiences. To address this, this section introduces a novel multi-dimensional media adaptation solution, which accounts for the uncertainties relating to unpredictable viewer and object behaviour as well as heterogeneous networking conditions. Machine learning is leveraged to account for behavioural and network awareness, with a multi-dimensional approach extending beyond quality-based adaptation introduced. Figure 15 shows and overview of the multi-dimensional adaptation solution. In this context, multiple facets are utilized to account for uncertainties associated with viewer, object and network behaviours. The viewer behaviour facet factors the viewer behaviour any subsequent expressed viewer intent in adaptation decisions. The network condition facet works in tandem with the viewer behaviour facet to ensure that all adaptation decisions are supported by the network. It also provides additional flexibility in adaptation by being able to adapt both rate and resolution. The behaviour of the source object is factored by the source object behaviour handling facet, allowing for relevant fps adaptations in response to the current object behaviour.

For example, if a viewer wearing a HMD while observing a holographic object was to move closer to the object, the viewer behaviour and expressed intent for a higher quality object representation, due to the relative viewer position, is accounted for via the viewer behaviour facet. After the expressed intent is confirmed, the relevant request is sent to the server. At the server, network information and other metrics are used with machine learning models to make a recommended resolution choice and subsequent adaptation request. Additional flexibility in decision making and adaptation with frame rate and resolution adaptation is also possible at the server to accommodate dynamic network conditions. Furthermore, the behaviour of the source object is also accounted for at the server, such that static object behaviour results in reduced fps adaptation and dynamic object behaviour enables increased fps adaptation. As such, if a viewer moves closer to the holographic object, the multi-dimensional adaptation solution would be able

to provide the highest possible resolution supported by the network and the most appropriate fps relative to the object behaviour, thus satisfying the viewer intent.

Figure 16 - Figure 18 shows the relevant FPS, throughput and quality representation performance of the proposed solution when handling diverse viewer and object behaviours. From the plots, it is evident that the solution can effectively accommodate diverse viewer and object behaviours. This is observed with the reductions in fps when the object becomes static, and the subsequent increase in fps when the object becomes dynamic. Furthermore, the increase in the resolution corresponds to the expressed viewer intent relative to the viewer position, as the resolution is increased when the viewer gets closer to the object and is reduced when the viewer is farther away. The multi-dimensional capability of the proposed solution is also evident when observing the fps plot as the fps is reduced when the object is static, but the resolution is increased at the same time when the viewer is close to the object. Similarly, when the object becomes dynamic, the fps is increased to the maximum supported value relative to the resolution level. The throughput also provides further confirmation on this, with relevant increments and decrements corresponding to the adaptation decisions. As such, it is evident that the multi-dimensional adaptation solution can satisfy the viewer intent whilst accounting for various uncertainties.

- The viewer behaviour is handled by leveraging a dedicated user intent module which allows for the accounting of viewer behaviour and registration of viewer intent.

- Machine learning is utilized for the handling of the dynamic nature of the network, ensuring that network aware recommendations are provided.

- The handling of the source object movement also leverages machine learning for the recognition of source object movement patterns and subsequent recommendations.

- The various facets are factored in the adaptation decision making algorithm, which utilizes all the acquired information to make a suitable adaptation request.

The combined utilization of these facets enables the algorithm to adequately account for viewer behaviour and expression of intent, source object behaviour and the dynamic nature of the network and their respective uncertainties. This ensures that recommendations and adaptation decisions which are made in real time are appropriate and optimal (i.e. reflective of the various behaviours whilst ensuring that the network capability is factored). As such the adaptation solution enables the satisfaction of viewer intent and immersive experiences.

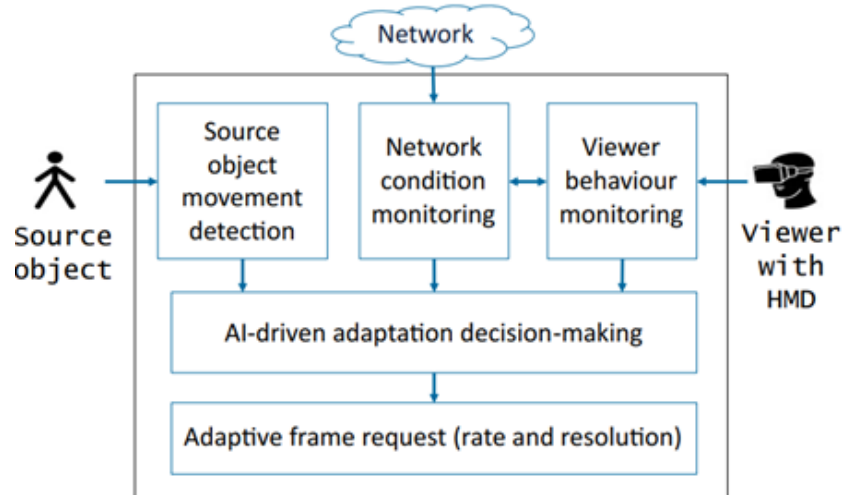Integration efforts on existing platforms are being investigated.

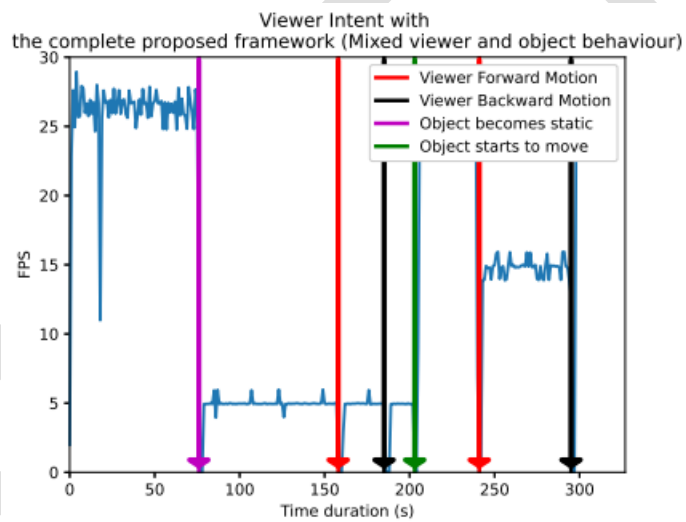*FIGURE 15 MULTI-DIMENSIONAL ADAPTATION SOLUTION FRAMEWORK*

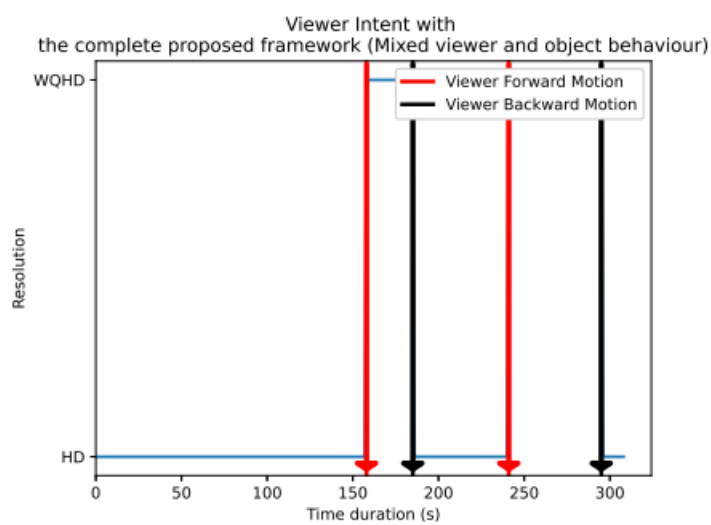*FIGURE 16 MULTI-DIMENSIONAL ADAPTATION SOLUTION PERFORMANCE (FPS)*

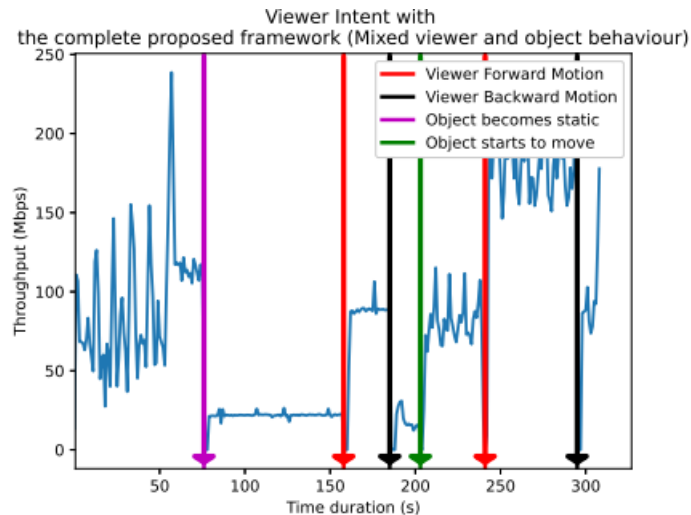*FIGURE 17 MULTI-DIMENSIONAL ADAPTATION SOLUTION PERFORMANCE (RESOLUTION)*

FIGURE 18 MULTI-DIMENSIONAL ADAPTATION SOLUTION PERFORMANCE (THROUGHPUT)

3.1.4 Frame Synchronisation for Multi-Source Holographic Communication

Holographic communication applications experience impediments limiting its adaptability and applicability, due to rather expansive network and computational requirements. This is further reflected in applications such as live multi-source holographic communication, which facilitates distributed/shared experiences on an internet scale, via the simultaneous teleportation of multiple source objects/people from distinct remote locations into common space. Moreover, these limitations are further heightened when compounded with the onset of motion misalignment which could be prevalent between source objects possessing different remote network profiles (i.e. latencies), representing an area of concern for viewer experiences. For example, in a multi-source scenario that presents a significant degree of motion between multiple source objects, motion misalignment is more likely to be perceived by the viewer, particularly when considering different network profiles, resulting in a potential degradation of the viewer experience.

As such, to address this, this section introduces a novel frame synchronization mechanism for multi-source holographic communication, aimed at limiting or avoiding motion misalignment between source objects. The findings have been published in [33]. In this regard, Figure 19 depicts a multi-source scenario with frame synchronization applied in a network environment.

Here, if a frame from Source1 is the first arrival, it is queued in a waiting queue. Then, the mechanism waits for a frame from Source2 with a similar/same timestamp mirroring Source1, determined by the Frame paring approximation threshold (γ), which provides an additional level of approximation in order to pair the frames together factoring small-time offset between their timestamps. If the succeeding frame arrives within the synchronisation window Δ which is the tolerable time window afforded for late frame arrivals, the algorithm checks the time gap between the frame pair, and forwards the frame pair if it is smaller than or equal to γ. Further, it caches the frame pair for future pairing in case the next expected frame does not arrive in time. If the synchronisation window expires, and there are no new frame arrivals from Source2, the algorithm searches for a cached frame from Source2 which is within the frame pair approximation threshold. If such a cached frame exists, the algorithm checks whether the time gap between the initial frame and the cached Source2 frame is within the γ threshold or not. If it satisfies the condition, then the initial frame is cached, and the frame pair is forwarded. Otherwise, the initial frame is dropped for pairing, but it is still cached.
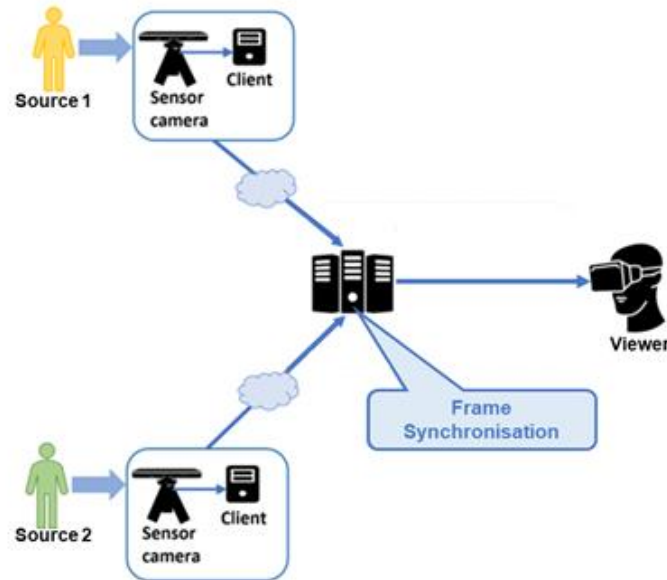
FIGURE 19 :MULTI-SOURCE SCENARIO WITH FRAME SYNCHRONIZATION

Results

The results depict the evaluation of the mechanism when considering the same FPS configurations for both sources. The key terminology regarding the frames is detailed below:

Fresh frames: Frames with the same timestamp that arrive at the MEC server within the synchronisation window (Δ) from all connected sources.

Half fresh frames - If a frame from one of the sources does not arrive within the synchronisation window, instead of discarding the frame, the system pairs a previously cached frame with the waiting frame.

Dropped frames - Frames that arrive at the MEC server which are neither fresh nor half-fresh frames are dropped, and are not used.

Figure 20 depicts the frame distribution with the Frame paring approximation threshold (γ) at a stringent value of 10ms and a varying synchronisation window from 5ms to 50ms. It is evident from the figure, that in the scenario considering the synchronisation window of 5 ms and 10 ms, the distribution of half fresh frames is almost equal to the proportion of fresh frames. This is due to the stringent nature of synchronisation window under such cases, ensuring that the wating period for frames from other sources is low. As such, after the expiration of the synchronisation window, the waiting frame searches and pairs with a qualified cached frame based on the approximation threshold. However, when considering a more lenient window of 50 ms, the proportion of fresh frames is only around 60%. This pattern is associated with the reduced FPS at the receiver side, as one of the sources is 50ms away. Thus, it is evident that stringent γ coupled with the reduced FPS affects the performance of the application. As compared to the 10 ms case, the number of fresh frames increase by 157% for the 50 ms case. For a synchronisation window between 30 ms to 50 ms, the proportion of fresh frames increases as the synchronisation window increases.
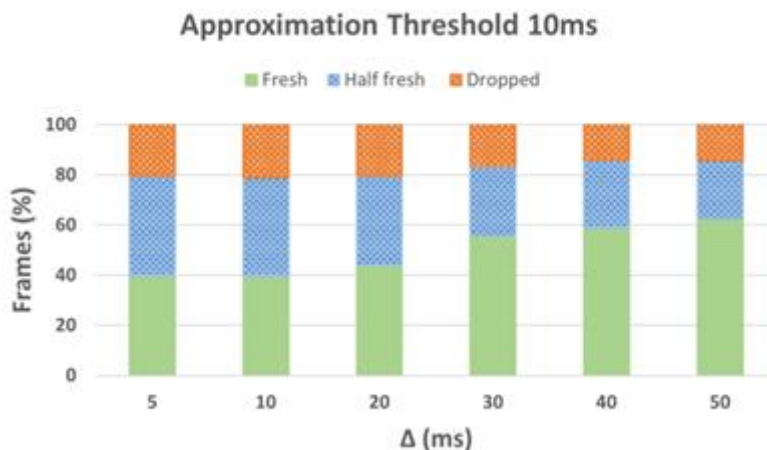
Approximation Threshold 10ms

FIGURE 20 FRAME DISTRIBUTION AT Γ =10MS

Figure 21 depicts the frame distribution with the Frame paring approximation threshold (γ) at a relaxed value of 50ms and a varying synchronisation window from 5ms to 50ms. It is evident that when compared to the strict case of γ =10ms, the proportion of fresh frames increases significantly. This pattern is attributed to the reduced sensitivity to the synchronisation window, due to the relaxation of γ. This is further reflected with slight fresh frames increases for higher values of Δ like 40 ms and 50 ms.

Integration efforts on existing platforms are being investigated.

Approximation Threshold 50ms
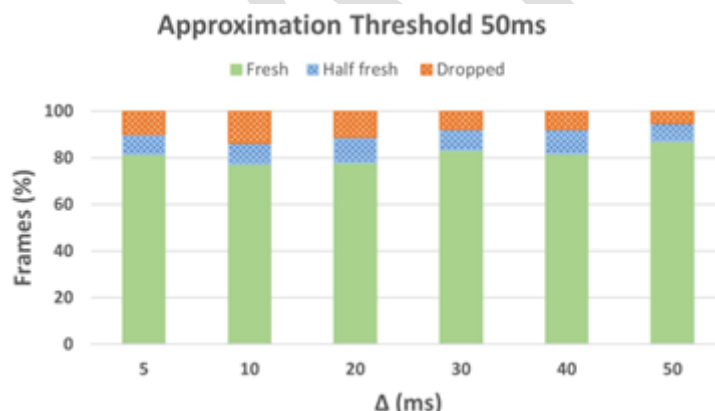
FIGURE 21 FRAME DISTRIBUTION AT Γ = 50MS

3.2 SCALABILITY FOR TELEPRESENCE APPLICATIONS

The current application platforms presented in Section 2.4 support one-to-one conferencing through their native WebRTC implementation. As illustrated in Figure 10 and Figure 12, intricate processing steps, including filtering and encoding, have been implemented to enhance volumetric content delivery for the one-to-one conferencing use case.

However, the objective of the SPIRIT project is to transcend limitations from one-to-one conferencing and venture into one-to-many and many-to-many conferencing to support scalability of the SPIRIT platform.

3.2.1 Volumetric Video Delivery for Real-Time Telepresence: One-to-Many Conferencing

Compared to D3.1, this section has been extended to discuss the architecture of our one-to-many volumetric video conferencing system in more detail, and to provide the most relevant results on the end-to-end latency. Our work on this topic has so far resulted in one published conference paper at ACM Multimedia Systems 2024 [34] and one journal paper submitted to ACM Transactions on Multimedia Computing, Communications, and Applications [35].

The adoption of immersive video capturing and rendering methods has been hindered by their substantial bandwidth and computational requirements, rendering them impractical for commercial applications. Several efforts have been made to alleviate these problems by introducing specialised compression algorithms and by utilising existing 2D adaptation methods, such as HAS to adapt the quality based on the user's available bandwidth. However, even though these methods help improve the QoE and bandwidth limitations, they still suffer from high latency which makes real-time interaction unfeasible.

To address this issue, this section presents a novel one-to-many streaming architecture using volumetric video based on point clouds. To reduce the bandwidth requirements, the Draco codec is utilised to compress the point clouds before they are transmitted using **WebRTC** which ensures low latency, enabling the streaming of real-time 6DoF interactive volumetric video. Content is adapted by employing a multiple-description coding (MDC) strategy which combines sampled point cloud descriptions based on the estimated bandwidth returned by the Google Congestion Control (GCC) algorithm. This allows us to more easily scale to a larger number of users, compared to performing individual sampling and encoding.
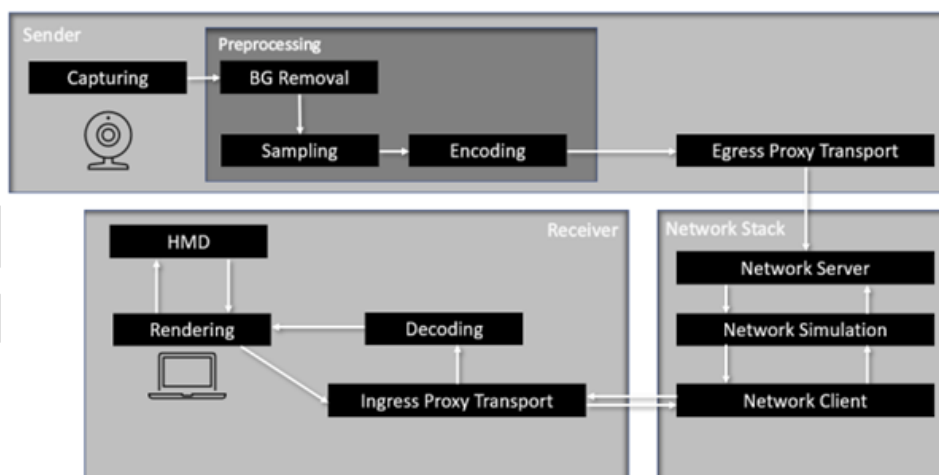
FIGURE 22 ARCHITECTURE FOR POINT CLOUD-BASED VOLUMETRIC VIDEO CONFERENCING

Figure 22 shows an overview of the end-to-end architecture, where several optimisations are proposed to enhance throughput and reduce latency, thus improving the quality of experience (QoE) to end users.

3.2.1.1 Architecture

To enable quality adaptation, we developed a video conferencing system that makes use of a centralized content server. The resulting system architecture is depicted in Figure 22. It contains the following components:

- **Capturing.** While the client can technically use any RGB-depth camera, our setup utilizes a single D455 Intel RealSense camera[10] with a frame rate of 30 FPS and a resolution of 848x480 pixels for each user. This camera captures both depth and colour images, which are transformed into a point cloud representation using a mapping function based on those included in the RealSense SDK[11]. The laser power of the camera is maximized to enhance the visual quality.

- **Preprocessing.** The raw point cloud frames produced by the camera are too large (1.4 Gb/s) to be transmitted directly over contemporary networks. Additionally, the camera also captures background details that are not needed for virtual conferencing. To reduce the required bitrate, the following preprocessing steps are executed:

  - A rudimentary distance filter removes points that are too far away from the camera.
  - The number of points in the point cloud is limited to a maximum, preventing sudden bandwidth increases between consecutive frames.
  - The point cloud is uniformly sampled into multiple descriptions, each containing a distinct subset of the points. In our setup, we used three descriptions containing 15%, 25% and 60% of the points.
  - The Draco codec[12] encodes the resulting descriptions in parallel.

  The MDC-based approach facilitates the creation of multiple combinable quality representations from one single captured object, allowing for a wider range of quality representations with differing bitrates. In our setup, this requires a fixed number of encoders per participant, corresponding to the number of generated descriptions.

- **Delivery.** We created a custom framework based on the Pion Golang package[13], an existing implementation of the WebRTC protocol suite. Pion makes it possible to transmit data via both data channels and media tracks. Although both methods could be adapted for point cloud video, the use of media tracks enables access to the congestion control feedback mechanisms implemented in Pion and allows complex bandwidth estimation using GCC. As the pipeline uses a novel frame format, a new WebRTC media track type is created to accommodate the differences in transcoding and packetization compared to traditional 2D video. To ensure that frames remain decodable in the event of packet loss, lost packets are retransmitted using a NACK-based solution.

  For one-to-many communication, a multipoint control unit (MCU) is used to synchronize incoming descriptions and concatenate them together in a single point cloud frame for every individual user. The generated frame is subsequently transmitted to the client over a single dedicated media track.

- **Decoding.** Unlike the encoding process, the number of received descriptions depends on the available bandwidth. For this reason, a thread pool is used with a number of concurrent workers. Whenever a worker becomes available, a decoding job is dequeued. Once the received descriptions of the point cloud video are decoded, they are merged together into a single point cloud object. The decoding operation utilizes the official Draco Unity plugin[14].

---

[10] https://www.intelrealsense.com/depth-camera-d455/

[11] https://github.com/IntelRealSense/librealsense

[12] https://github.com/google/draco

[13] https://github.com/pion/webrtc

[14] https://github.com/atteneder/DracoUnity

- **Rendering.** Unity[15], a game engine commonly used to develop video games, is used to design the application responsible for rendering the point clouds. The OpenXR plugin[16] included within Unity enables the use of several head-mounted displays (HMDs). In our setup, the Meta Quest 2 headset[17] is used in conjunction with the Meta Quest Link application, allowing us to stream the rendered frames to the Meta Quest headset. Visualizing the point cloud in the application is accomplished through the Pcx package[18]. By leveraging the shader included in the package and applying it to a mesh component, Unity renders the vertices of the mesh as individual point primitives. As a result, the vertices can be set at runtime to achieve dynamic point cloud rendering.

3.2.1.2 Results

The proposed pipeline has been used to run experiments locally and on the Virtual Wall infrastructure. This allowed us to highlight the most important advantages compared to the state of the art. As an example, Figure 23 shows the bitrate ladders of an example point cloud video for fixed quality levels and the suggested MDC-based approach. This shows that we can achieve higher granularity with fewer encoders, which is expected to result in a better user experience. This is illustrated in Figure 24Figure 23**Error! Reference source not found.**, which shows the used bandwidth (video bitrate) as a function of the available bandwidth (applied 4G bandwidth trace [36]). When three fixed quality representations are available, the available bandwidth is mostly underused, since the next target on the bitrate ladder is too high. This is not the case for seven fixed quality representations, nor for three descriptions with MDC.
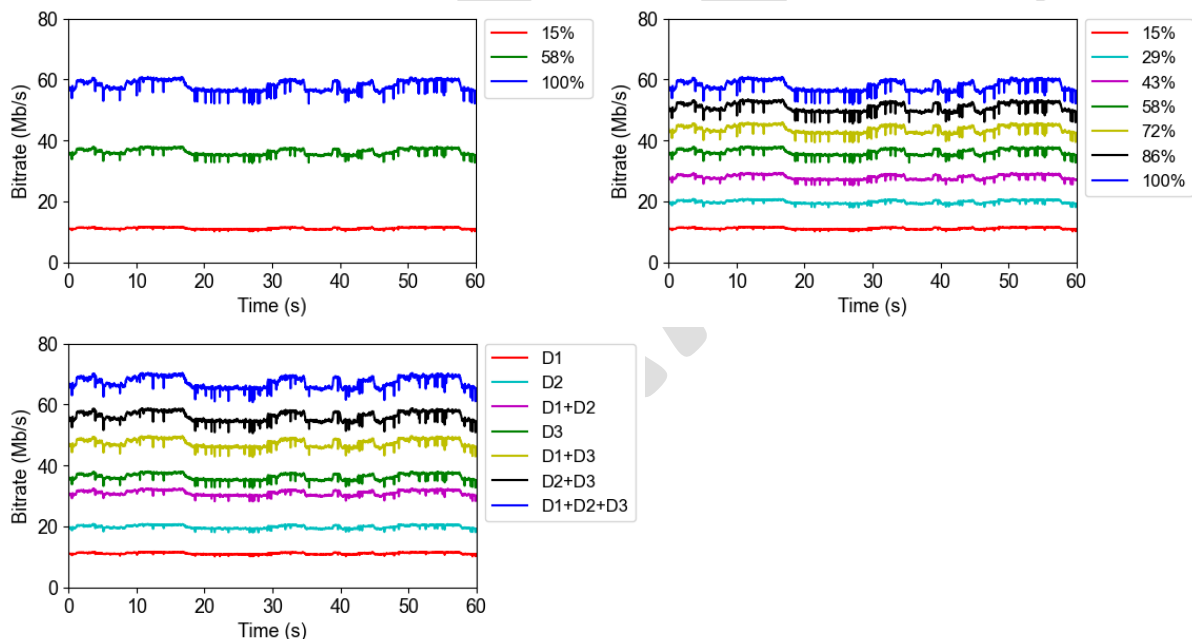

FIGURE 23: BITRATE LADDERS FOR 3 FIXED QUALITY LEVELS (TOP LEFT), 7 FIXED QUALITY LEVELS (TOP RIGHT) AND MDC WITH 3 DESCRIPTIONS (BOTTOM)

---

[15] https://unity.com/

[16] https://docs.unity3d.com/Packages/com.unity.xr.openxr@1.11/manual/index.html

[17] https://www.meta.com/be/en/quest/products/quest-2/
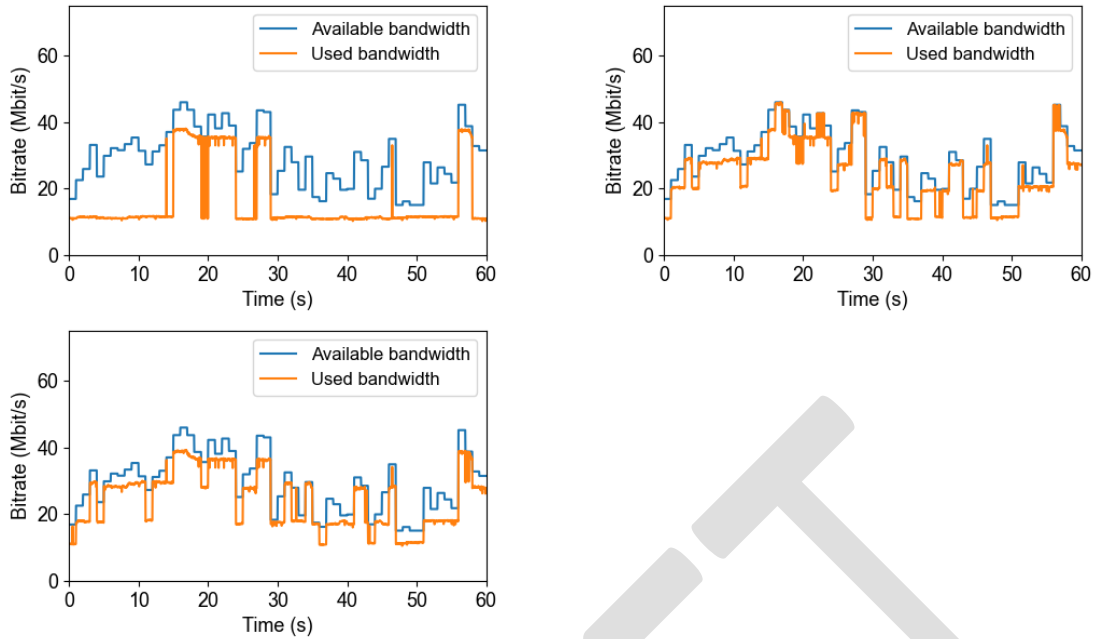
[18] https://github.com/keijiro/Pcx

FIGURE 24: BANDWIDTH ADAPTABILITY FOR 3 FIXED QUALITY LEVELS (TOP LEFT), 7 FIXED QUALITY LEVELS (TOP RIGHT) AND MDC WITH 3 DESCRIPTIONS (BOTTOM)

Providing a bitrate ladder with higher granularity results in a higher video quality, as illustrated in Figure 25**Error! Reference source not found.**. While differences are not statistically significant, the average VMAF scores[19] are higher when more quality levels are provided. Furthermore, results for the proposed MDC approach closely resemble those achieved for 7 fixed quality representations.
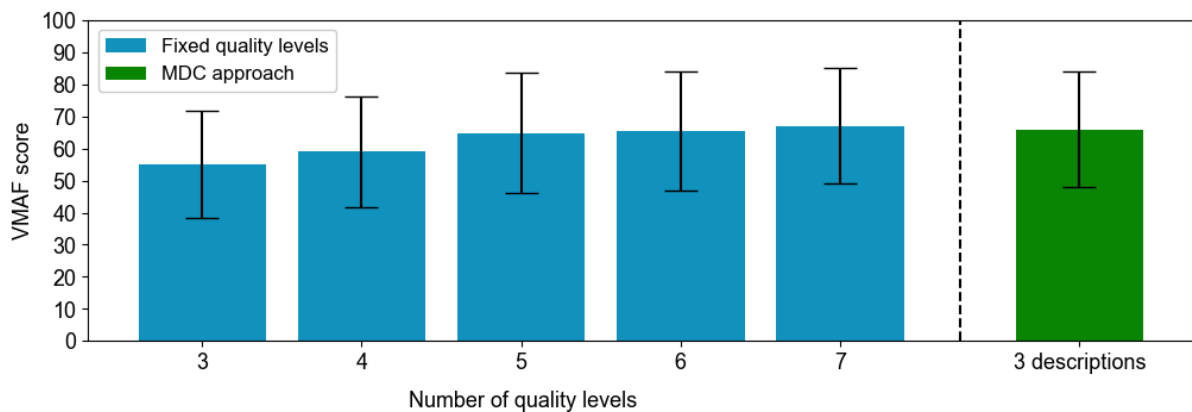
FIGURE 25: VMAF SCORES FOR FIXED QUALITY LEVELS AND MDC

Since we consider real-time immersive video conferencing, latency aspects should also be considered. Figure 26**Error! Reference source not found.** shows the encoding time needed to generate a fixed number of quality layers, using multithreading with three different workers. Naturally, increasing the number of representations results in a higher encoding time. Our MDC approach outperforms this approach even when the number of descriptions equals the number of fixed quality layers, because subsampled versions of the point cloud object are considered. This results in a significantly lower encoding time. Finally, Figure 27 shows the end-to-end latency

---

[19] https://github.com/Netflix/vmaf

(i.e., from capturing to display) for three configurations. While capturing takes the same time for all configurations, using our MDC approach results in an end-to-end latency that is slightly higher than for 3 fixed quality levels, but significantly lower than for 7 fixed quality levels. The preprocessing step (which includes background removal and encoding) is shorter when compared to fixed quality levels, while network transport takes slightly more time.
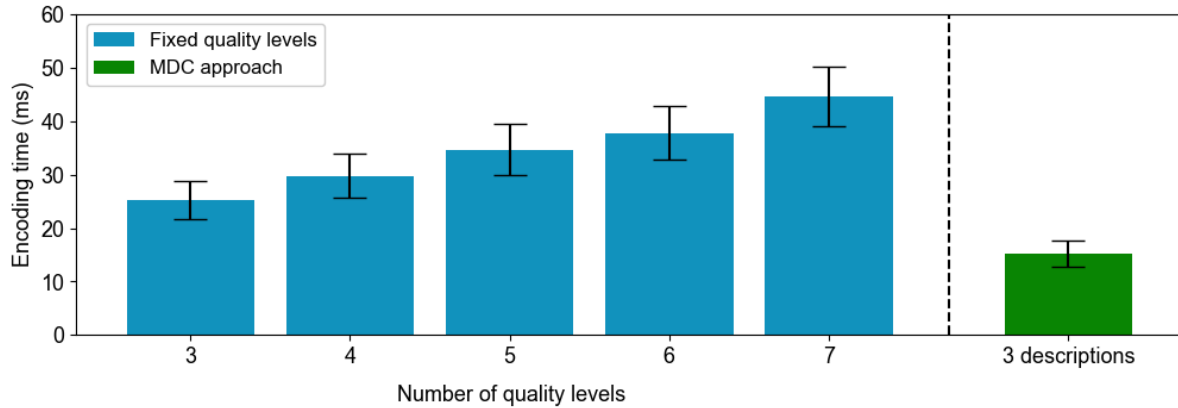
FIGURE 26: ENCODING TIME FOR FIXED QUALITY LEVELS AND MDC
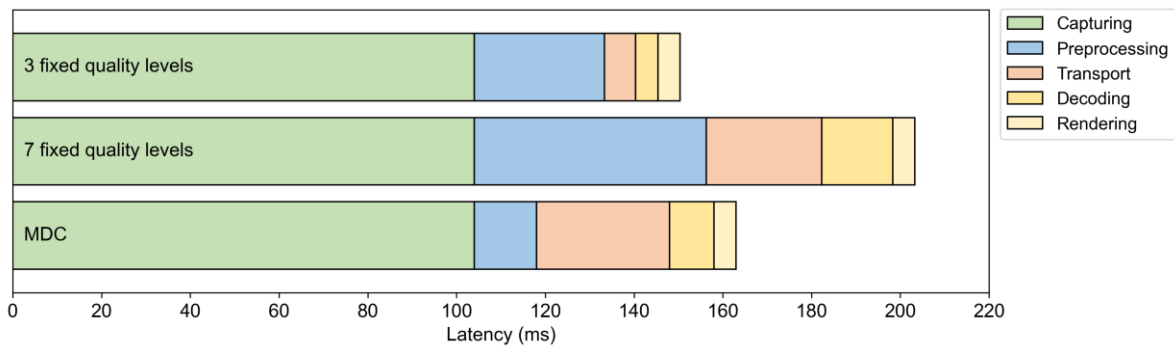
FIGURE 27: END-TO-END LATENCY FOR FIXED QUALITY LEVELS AND MDC

For a more detailed discussion of the conducted experiments and the full results in terms of latency and visual quality, we refer to [34, 35].

Code availability

All code for one-to-many[20] volumetric video delivery has been made available online with detailed instructions on how to install and use the software.

3.3 VOLUMETRIC VIDEO DELIVERY FOR REAL-TIME TELEPRESENCE: MANY-TO-MANY CONFERENCING

Compared to D3.1, this section has been extended to discuss the architecture of our many-to-many volumetric video conferencing system in more detail, and to provide preliminary results on

---

[20] https://github.com/idlab-discover/pc-webrtc-o2m

the end-to-end latency. Our work on this topic has so far resulted in one published demo paper at ACM Multimedia Systems 2024 [37].

In immersive multi-party conferencing, the user is envisioned to wear a head-mounted display, rendering an immersive video scene where each peer is represented through an avatar that mimics the peer's movements based on camera or sensory data. Rather than using avatars, however, recent attempts consider volumetric video to represent the different peers, captured through low-complexity RGB-depth cameras. To deliver the content between peers, systems consider web sockets, which are not suitable for scalable delivery, or low-latency DASH, which results in an end-to-end delay in the order of one second [30, 38].
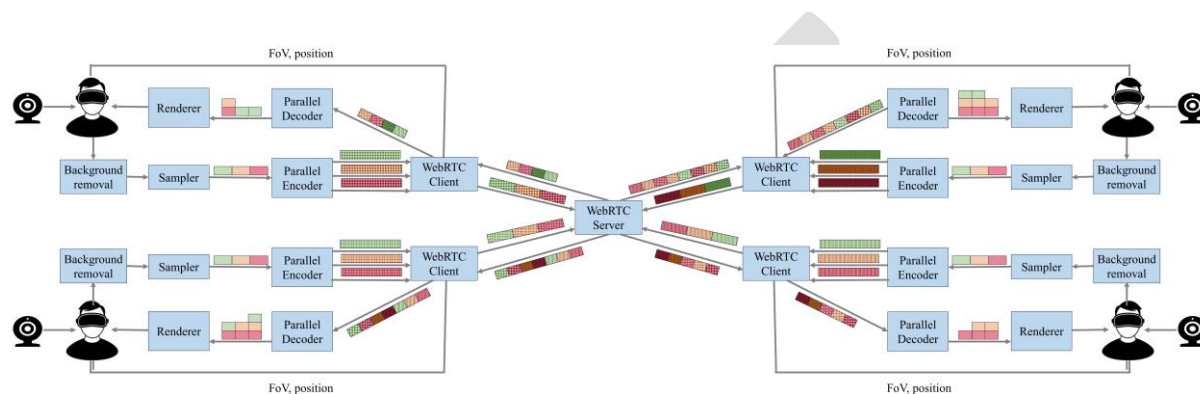

FIGURE 28: SFU-BASED WEBRTC DELIVERY

To address this issue, this section presents a WebRTC-based many-to-many streaming architecture for volumetric video, illustrated in Figure 28. Compared to our one-to-many approach, the following changes have been made:

- **Adoption of an SFU.** While we considered an MCU approach for one-to-many volumetric video delivery, the proposed multi-party system makes use of a selective forwarding unit (SFU) that receives incoming streams from all participants These streams typically carry a single point cloud or mesh, which represent a single frame of the captured end user. The SFU selectively forwards these streams to clients that require them, considering the parts of the content that are currently visible to the consumers. To this end, all participants use periodic signalling to inform the SFU in real time on what parts of the scene are currently within the user's field of view (FoV); any streams related to content outside of the FoV can then simply be discarded.
- **Quality adaptation**. The concept of quality adaptation is a well-established practice in conventional streaming to ensure smooth video playback. Most encoders generate several quality representations by supplying varying target bitrates [12]. However, for a larger number of users, it becomes unfeasible to have individual encoding. In the proposed system, the quality adaptation is based on an MDC-based approach, which combines the descriptions generated in the preprocessing step to obtain a broader spectrum of available qualities with differing bitrates. In general, n descriptions result in $2^n - 1$ possible combinations to choose from.
  Using the estimated bitrate obtained from the GCC algorithm, together with the position and FoV of each user, quality adaptation takes place on the central server. The applied algorithm aims for a fair approach which searches for a balance between assigning a high quality to nearby client frames and a lower quality to those further away. The distance between the frame and the receiver's position determines the maximum assignable quality level, as illustrated in Figure 29.

FIGURE 29: A POINT CLOUD'S QUALITY CATEGORY IS ASSIGNED BASED ON DISTANCE AND FOV

In the first step, the algorithm uses an iterative approach that assigns available bandwidth to all objects, starting from the furthest object in the FoV. The assigned bitrates are calculated by uniformly distributing the available bandwidth with respect to the maximum assignable quality, considering the number of objects in a specific category (see Figure 29). Each object is assigned at least the minimal quality, even if there is not enough bandwidth available to accommodate all objects. In a second step, the bandwidth allocation process starts from the opposite direction and reallocates bandwidth from objects further away to those that are closer, until there is enough bandwidth to stream the most important objects at least at the minimal quality. We refer to [34] for more details on the adopted quality adaptation algorithm.

- **Rendering.** When it comes to rendering, a similar approach to the one-to-many use case is adopted. In this case, however, multiple objects are received, processed and rendered in different locations in the virtual environment. Furthermore, the rendered size of points belonging to sparser objects (i.e., those objects that did not receive all descriptions) can be increased to increase the resulting video quality.

Both single- and multi-tile delivery is supported for many-to-many conferencing. The SFU has been extended to enable per-tile decision-making and delivery, taking bandwidth and latency efficiency into account.

3.3.1.1 Results

Currently, evaluations are planned to determine the resulting end-to-end latency and video quality through large-scale network emulation.

3.3.1.2 Demonstrator

As part of a demonstrator at ACM Multimedia Systems 2025, a web-based dashboard was designed to visualize relevant data in real time [37]. An illustration is shown in Figure 30, showing five different panels:

1. Illustrates the FoV of each client.
2. Indicates through colour/numerical value which quality level is received for each client.
3. Shows the used bandwidth and estimated bandwidth, as well as a graph illustrating the bandwidth trace for each user.
4. Shows which clients are visible for each participant, as well as having the ability to change the position of each participant in the virtual environment.
5. Is used to change the bandwidth available to a client.

This dashboard not only allows us to monitor the current quality, latency and bandwidth usage for each user, but also to change parameters such as the available bandwidth and the network

latency. During the session, the user can freely move their head and body; when the user's FoV changes, the WebRTC server updates the selected video qualities accordingly.
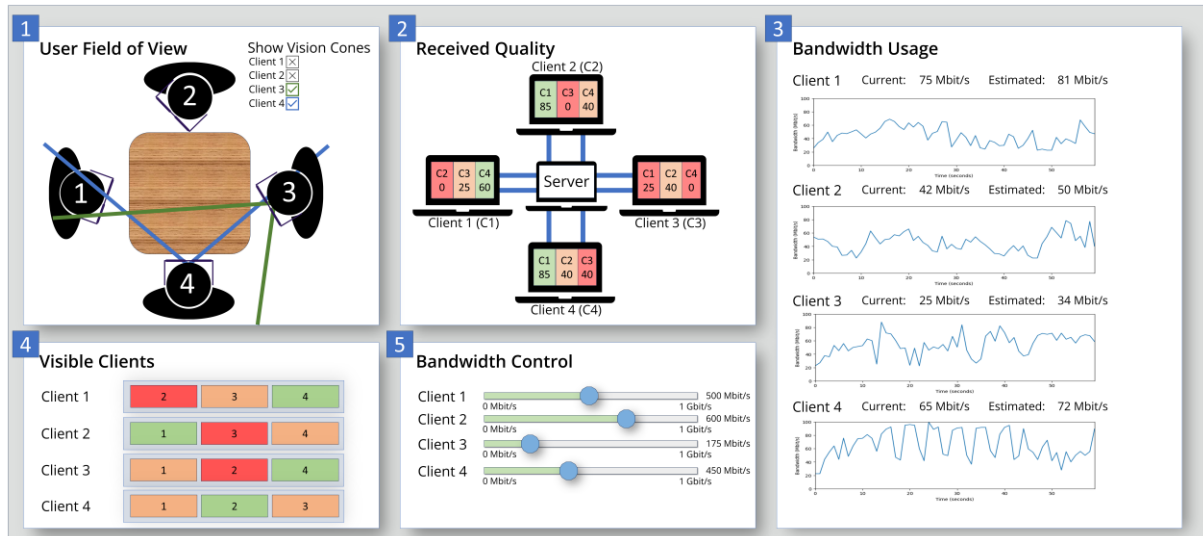

FIGURE 30: ILLUSTRATION OF THE DASHBOARD USED DURING THE DEMONSTRATION

3.3.1.3 Code availability

All code for many-to-many[21] volumetric video delivery has been made available online with detailed instructions on how to install and use the software. Integration efforts on the platforms are being investigated.

3.4 SPLIT RENDERING WITH USER INTERACTION FOR TELEPRESENCE APPLICATIONS

Streaming volumetric objects through a network in real time presents several technical hurdles, the main one being the magnitude of the data load to be sent, especially when handling realistic representations of humans, where the complexity of the geometry is significant. On top of this, the already mentioned amount of data varies from object to object, making it difficult to assess what the exact requirements for the network must be.

To overcome this, a Split Rendering mechanism has been implemented. The object pool, implemented in the server, is a component that handles the position, rotation, and scale of the object (avatar) at all times. At the same time, the client application sends messages continuously through the data channel, informing the server about the viewpoint of the user (position and orientation of the mobile device/XR glasses). A virtual camera situated in the Unity scene on the server follows these movements and renders a 2D image from the appropriate position. This results in a different view of the avatar each time. This process is illustrated in Figure 31 for the photorealistic avatar use case.

---

[21] https://github.com/idlab-discover/pc-webrtc-m2m
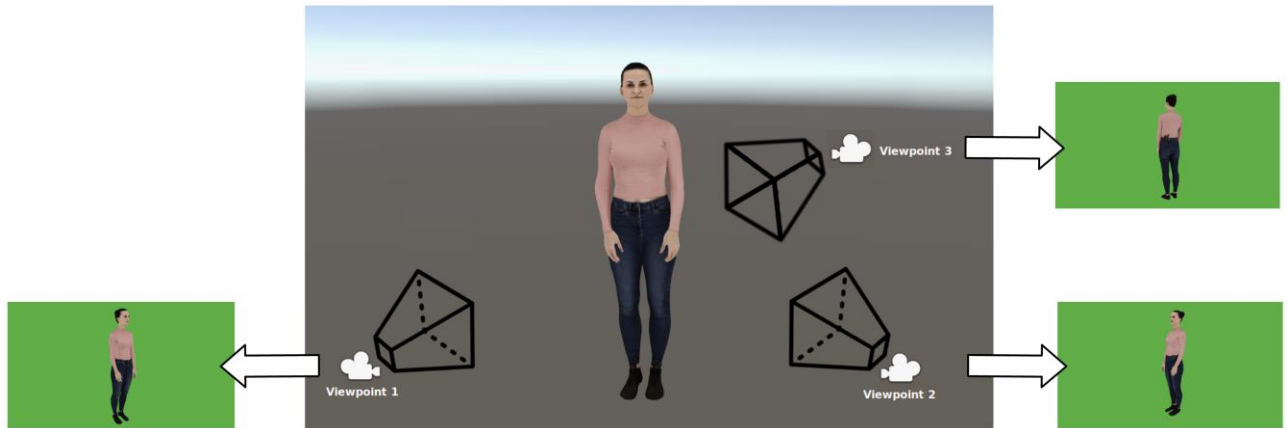
FIGURE 31: RENDERING PROCESS ON THE SERVER SIDE

This outcome is a single image, or an image formed as a combination of two camera renders, placed side by side, in the stereoscopic case. The background is monochromatic.

The rendered image is then encoded in a Gstreamer[22] pipeline and sent through the network using WebRTC.

When the client receives the image, the second part of the rendering process takes places. This time, the client application removes the monochromatic background (whose specific colour is known by both server and client) and provides the player module with an image where the avatar is integrated in the real scene (Figure 32).

Once the consumer user can see and hear the avatar, certain interaction functionalities, such as dragging and dropping the avatar, become available. In case of the tablet/mobile phone player, rotation and scaling can be performed as well. After each one of these actions, the new parameters (position, rotation, scale) of the avatar are sent to the server, so it can act accordingly.

---

[22] https://gstreamer.freedesktop.org/

FIGURE 32: INTEGRATION OF THE AVATAR IN THE SCENE

Split Rendering has been successfully implemented into the Real-Time Animation and Streaming of Realistic Avatars platform described in Section 2.4.2.

3.4.1 One-to-many Split Rendering

The original configuration envisioned for the application of Split Rendering techniques in telepresence scenarios aims only one-to-one communications where one user acts as generator of media and another one receives audio and video signals with which he/she can interact.

The next logical step is to extend this functionality to communication setups in which more than one person can receive individual video signals depending on his/her viewpoint. This implies several technical challenges, mainly on the management of the different rendering and streaming pipelines. The interaction between one of the user and the avatar (i.e. modifying its position) must be visible for the rest of the users. The streams will be synchronized on the server, but different scenarios with multiple users will be tested in order to assert how the latency affects the joint experience for the users and if additional synchronization mechanisms are required. Figure 33 shows this new architecture.
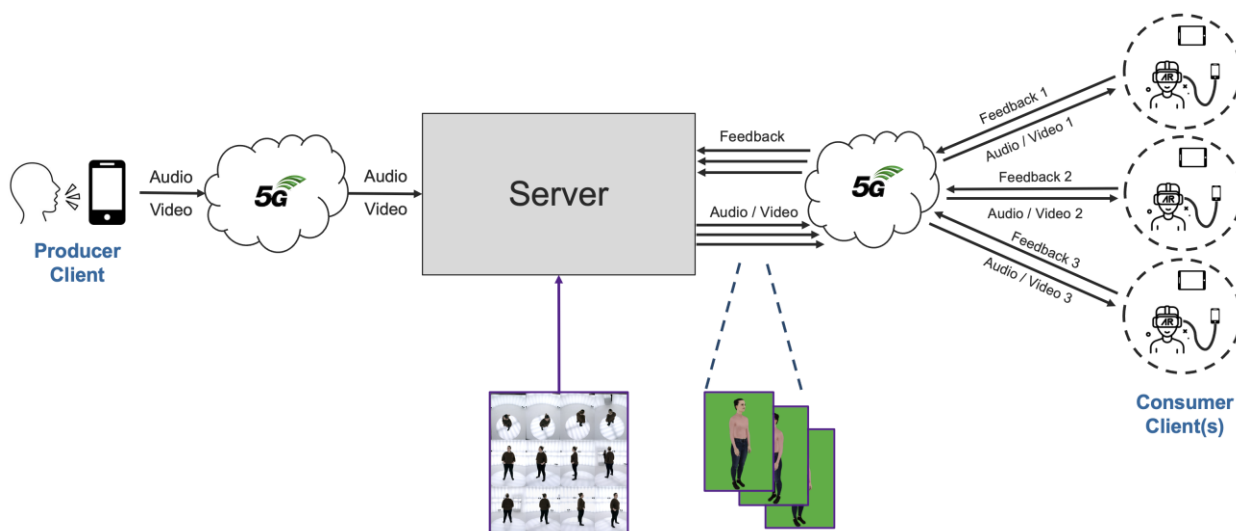
FIGURE 33 ONE-TO-MANY SCENARIO

Since the image rendered and sent by the server at each frame of the video is dependent on the real position of the user with respect to the avatar (viewpoint), the inclusion of different consumer clients requires the management and synchronization of different video signals and feedback data streams. In order to achieve this, new virtual cameras will be added to the Unity scene, each one representing one user. The images rendered by each camera will be then encoded in a dedicated Gstreamer pipeline and sent to the appropriate client by using separate WebRTC channels. The limitations related to the parallel processing capabilities of hardware encoders such as the Nvidia NVENC that is currently being used in this use case present themselves as one of the key points to achieve smooth streaming channels and to avoid extra latency. The initial goal of this one-to-many scenario is to have three different users connected at the same time. Nevertheless, the work will focus on step-by-step optimisations in order to achieve this planned performance.

The main goal of this extension is to open up the possibilities of the Real-Time Animation and Streaming of Realistic Avatars use case to make it available to new scenarios that require the participation of multiple users at the same time, such as educational and training environments.

The scalability of the current solution will be also tested with this new feature, obtaining valuable information on the behaviour of the system in more complex real-life situations.

3.5 SECURITY FOR TELEPRESENCE APPLICATIONS

3.5.1 Introduction to Security Challenges

Telepresence systems introduce unique security challenges that arise from the distributed nature of these systems. Cloud-based deployments add another layer of complexity to these issues. Additionally, there are identity and trust problems specific to telepresence applications as well as specific attack patterns. These subjects will be addressed in more detail in the next subsections.

3.5.1.1 Problem Statement: Cloud-based Telepresence Systems

The implementation of the SPIRIT system (and thus the utilisation of Edge Computing resources or the Public Cloud) presents new challenges in terms of security. The transmission of confidential data to the cloud can raise privacy concerns, especially when processing is carried out by foreign

cloud providers. An insecure network can enable hacker attacks that may jeopardise the integrity of the event. Technical issues or even Denial-of-Service attacks can lead to availability problems that can affect participation in the event. Additionally, it is crucial to control access to sensitive data and resources to ensure that only authorised individuals can access them.

Taking measures to address these security issues is essential to ensure that SPIRIT sessions can be conducted successfully and securely. In this document, we will primarily focus on the aspect of securing processing in the cloud.

When workloads are moved from on-premises 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 arise when deploying workloads on edge devices: again, the device may be under the control of an edge provider (e.g., a telecommunications operator) or deployed in exposed locations where it can be physically compromised by malicious actors (e.g., in publicly accessible conference rooms or exhibition halls).

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 integrity and confidentiality guarantees.

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 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 yet for widespread 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 [39].

Figure 34 presents a novel approach: end-to-end security, where data in transit is encrypted and only provably trust-worthy cloud applications can break up this encryption to process and store the confidential data. To exclude the cloud provider itself from the 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.
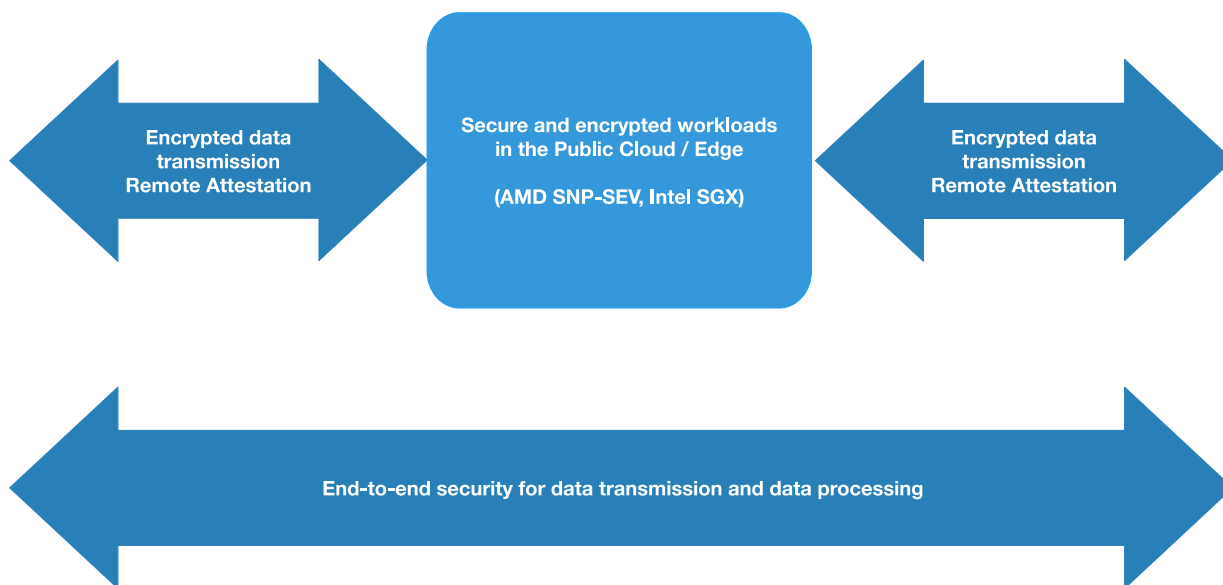
FIGURE 34: END-TO-END SECURITY

As Figure 34 implies, we will introduce the technologies developed by two main CPU manufacturers, Intel and AMD, which claim to provide these security guarantees. The rationale behind this decision is that security technologies offered by CPU vendors are not under direct control of cloud providers. This enables a separation of concerns. Since the development of trusted, open-source hardware is still in its infancy (and RISC V is a promising candidate here), one ultimately must trust at least the CPU vendors.

We will also look at the implementations of confidential computing offered by leading cloud providers – Microsoft Azure, Amazon Web Services, and Google Cloud – and will compare and evaluate them according to the goals outlined above.

Furthermore, some existing products and open-source projects will be introduced that aim to make packaging applications for secure edge or cloud deployments easier.

3.5.1.2   Previous Work on Telepresence Security

Data security in virtual and hybrid events has become a particular challenge due to the COVID-19 pandemic, as the number of cybercrime cases has risen. Sensitive data, such as payment and account information, intellectual property, and personal information, are exchanged during such events and are popular targets for hackers. Although online event platforms are often equipped with security mechanisms, there is still a risk of data breaches. Malware attacks and phishing are the most common threats to data security.

For the conduct of such events, there are several "standard" best practices that significantly enhance security:

1. One of the most important rules for hybrid and virtual events is to know exactly "who is participating". Event planners must be clear about who is participating, who these individuals are, and how many are participating. This way, you can prevent unwanted individuals and participants who do not belong to an event.

2. Participants involved in events bear significant responsibility for data security. They must adhere to basic security protocols and ensure that software and systems are updated to reduce the attack surface of their devices.

3. Robust network security is essential for the secure operation of the system. This includes monitoring, protecting, and responding to the system and ensuring secure access from external networks. This is particularly important for virtual events where participants and speakers are connected online from separate locations.

4. Organisers should prioritise platforms with end-to-end encryption for data transmission when securing virtual events. This technique makes data on the transmission channel unreadable until it reaches its destination.

5. The use of multi-factor authentication further enhances security. Participants are required to confirm their login credentials with an additional method, regulating access to the event and keeping unwanted participants, hackers, and disruptors at bay. Additionally, this measure can prevent disruption from DDoS attacks.

Following these approaches can significantly enhance the security of a virtual or hybrid meeting. However, the issue remains that data may be unencrypted on cloud platforms and could potentially be intercepted by malicious actors.

If data processing in the cloud is not necessary, there are practical approaches, e.g.: the Signal Protocol (formerly known as the TextSecure Protocol) is a non-federated cryptographic protocol that can be used for end-to-end encryption of voice calls and instant messaging conversations. It was developed by Open Whisper Systems in 2013 and first introduced in the open-source TextSecure app, which later became Signal. In addition to the open-source Signal Messenger, several closed applications have implemented the protocol, such as WhatsApp or Google, which provides end-to-end encryption by default for all RCS-based conversations between users of its Messages app for one-on-one conversations. Facebook Messenger and Skype offer the protocol for optional "Secret Conversations."

The protocol combines the Double Ratchet algorithm, prekeys, and a Triple Elliptic-curve Diffie-Hellman (3-DH) handshake using Curve25519, AES-256, and HMAC-SHA256 as primitives.

3.5.2 Overview of Trusted Execution Technologies for Cloud and Edge Workloads

In this section we introduce the concept of trusted remote execution the context of cloud and edge computing.
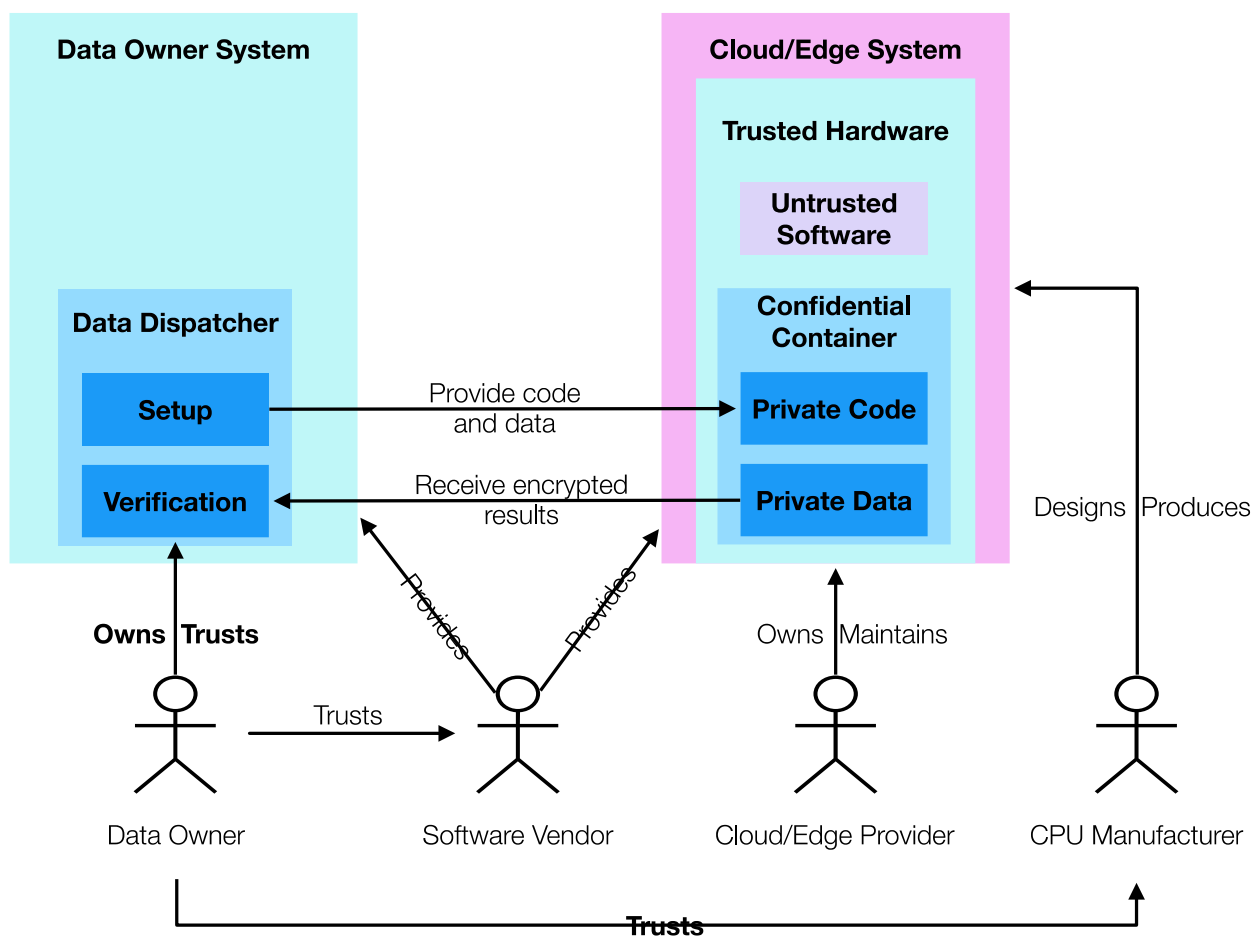
3.5.2.1 Trust Model

FIGURE 35: TRUST RELATIONSHIP IN CONFIDENTIAL COMPUTING

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

Figure 35 shows how the users trust the manufacturer of a piece of hardware in the remote computer and entrust their data to a secure container hosted by the secure hardware.

3.5.2.2 Hardware Solutions

Over the years several hardware solutions have been proposed and implemented by CPU manufacturers to make their respective computing platforms more secure.

On the X86/X64 side Intel has introduced the TXT platform [TXT] to integrate security functions into their processors. This was later followed by Software Guard Extensions (SGX) [40], which basically supersedes TXT technology as the built-in processor security platform. Over many evolutions of SGX, the security guarantees were extended such to enable secure workloads in the public cloud – a feature that will be covered in this document.

AMD, on the other hand, has designed the AMD Secure Processor (known as Platform Security Processor (PSP)), and developed over several iterations it into the SEV-SNP technology. This is also suitable for protecting cloud workloads and will be covered in this document [41].

Furthermore, separate security hardware anchors such as the Trusted Platform Module (TPM) and lately the Microsoft Pluton chip are integrated into server platforms. These hardware elements will be covered in this document only to extent that they are relevant for the Intel SGX and AMD SEV-SNP approaches.

Recently, Intel added a technology called Trust Domain Extensions (TDX) to protect and encrypt whole VMs similar to AMD SEV. We will cover this technology on a theoretical level, since TDX-enabled hardware is not readily available yet.

Moreover, ARM has IP building blocks using the brand name TrustZone, which implement hardware security features in ARM processors [42]. These IP building blocks are, however, optional for CPU manufacturers to integrate and focus more on Internet of Things -and mobile (Android and Apple mobile platforms) use cases. Even though ARM processors make inroads into the data centre, ARM does not have a security platform with TrustZone that is competitive with the AMD and Intel offerings. Therefore, we will not cover ARM TrustZone.

Intel SGX

Intel's SGX is a set of extensions to the Intel architecture that aims to pro- vide integrity and confidentiality guarantees to security-sensitive computations performed on a computer where all the privileged software (kernel, hypervisor, etc.) is potentially malicious.

Overview

SGX relies on software attestation, like its predecessors, the TPM [43] and TXT. Attestation (Figure 36) proves to a user that he is communicating with a specific piece of software running in a secure container hosted by the trusted hardware. The proof is a cryptographic signature that certifies the hash of the secure container's contents. It follows that the remote computer's owner can load any software in a secure container, but the remote computation service user will refuse to load his data into a secure container whose contents' hash does not match the expected value.
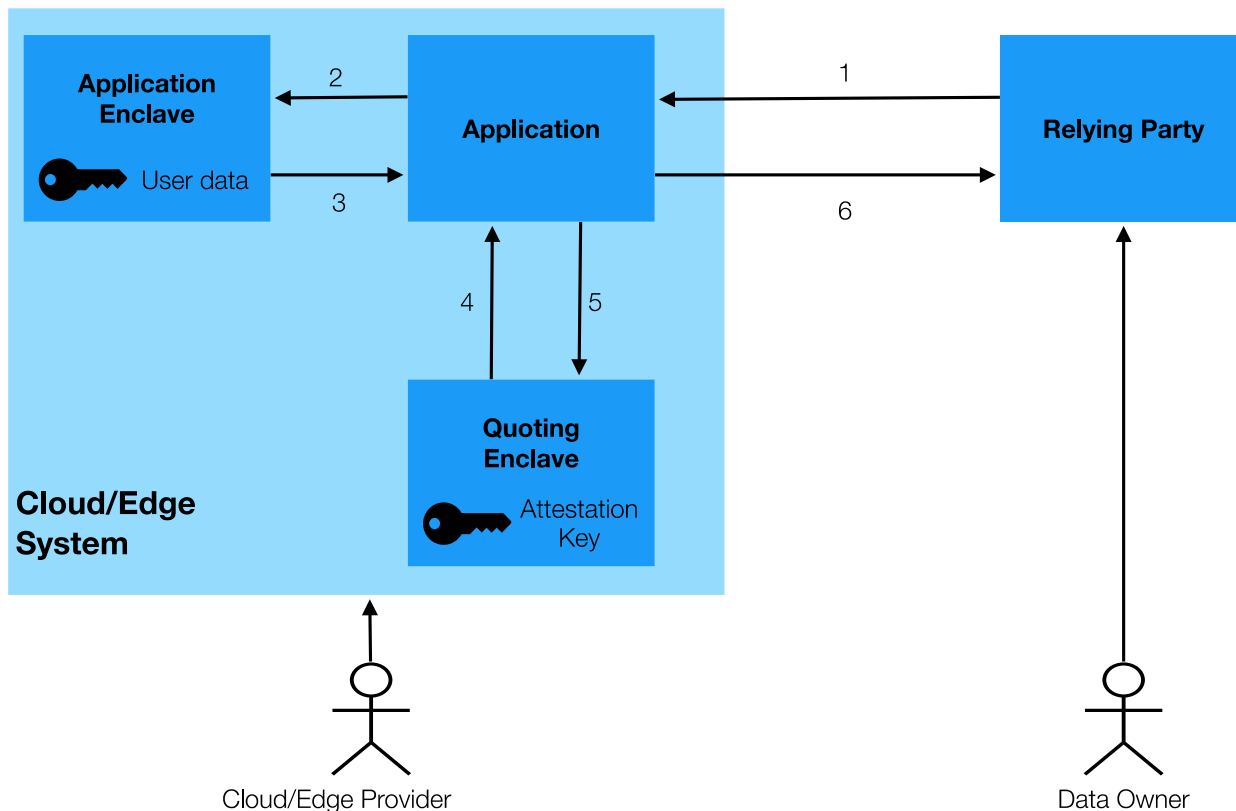
FIGURE 36: INTEL SGX ATTESTATION FLOW

This way, the user has some assurance of the computation's integrity and confidentiality before he uploads the desired computation and data into the secure container. The trusted hardware protects the data's confidentiality and integrity while the computation is being performed on it.

SGX stands out from its predecessors by the amount of code covered by the attestation, which is in the TCB for the system using hardware protection. The attestations produced by the original TPM design covered all the software running on a computer, and TXT attestations covered the code inside a virtual machine. In SGX, an enclave (secure container) only contains the private data in a computation, and the code that operates on it. For example, a cloud service that performs image processing on confidential medical images could be implemented by having users upload encrypted images. The users would send the encryption keys to software running inside an enclave. The enclave would contain the code for decrypting images, the image processing algorithm, and the code for encrypting the results. The code that receives the uploaded encrypted images and stores them would be left outside the enclave.

An SGX-enabled processor protects the integrity and confidentiality of the computation inside an enclave by isolating the enclave's code and data from the outside environment, including the operating system and hypervisor, and hardware devices attached to the system bus. At the same time, the SGX model remains compatible with the traditional software layering in the Intel architecture, where the OS kernel and hypervisor manage the computer's resources.

According to the SGX patents, all the SGX instructions are implemented in microcode. The SGX patents state that SGX requires very few hardware changes, and most of the implementation is in microcode. This allows Intel to distribute feature and security updates to SGX as microcode updates while decreasing the number of architectural changes to the Intel platform.
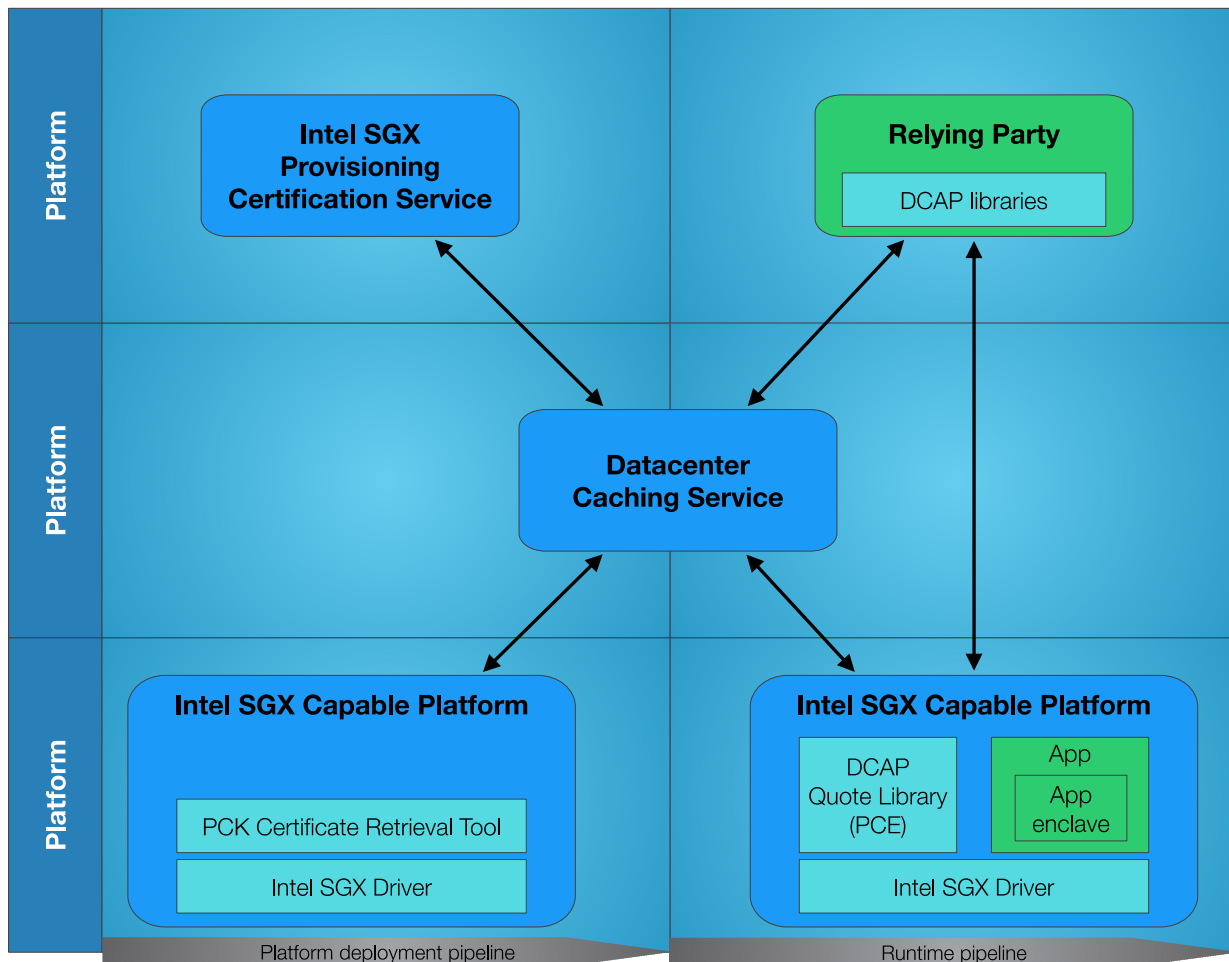
FIGURE 37: INTEL SGX PLATFORM ARCHITECTURE (DATACENTRE).

Figure 37 shows a high-level overview of the Intel SGX deployment and runtime pipeline as well as the most important architectural elements.

When it comes to the platform, both the development and the runtime systems must support Intel SGX in the CPU hardware and have the necessary SGX drivers installed. The Provisioning Certification Key (PCK) – unique to the specific CPU hardware – is available to the Provisioning Certification Enclave (PCE) provided by Intel. The PCE can sign proofs in the context remote attestation. The certificate for the PCK can be retrieved from Intel SGX Provisioning Certification Service. To minimise dependencies from this Intel-run service (and for privacy/data protection reasons) a datacentre operator may decide to run a local Datacentre Caching Service for these certificates. The infrastructure for this datacentre caching service is called Data Centre Attestation Primitives (DCAP).

When deployed, an SGX application consists of two parts: A normal application program, running as a user program on the target hardware, and an SGX App Enclave that is protected by the SGX security guarantees. A relying party (communication partner) only transmits confidential information to the SGX App Enclave after successful remote attestation. Intel provides software support in the form of SDKs and libraries to support the cryptographic operations and the communication with the Intel provisioning service to retrieve, e.g., PCK certificates. This infrastructure also supports caching on the relying party side to minimise dependencies on Intel services.

While the Intel SDKs for Linux and Windows are free of charge, they only allow development of SGX software in debug mode. To use SGX in production mode, developers must request a commercial license from Intel. The license is currently free of charge, but the process requires approval from Intel [44].

Practical Experiences

Refer to "A.1.1 Practical Experiences with Intel SGX" for details on our experimentation.

Known Attacks

The research community has uncovered some attacks on Intel SGX that break the security guarantees mentioned in the previous section:

- 2018: „Foreshadow is a speculative execution attack on Intel processors which allows an attacker to steal sensitive information stored inside personal computers or third-party clouds. Foreshadow has two versions, the original attack designed to extract data from SGX enclaves and a Next-Generation version which affects Virtual Machines (VMs), hypervisors (VMM), OS kernel memory, and System Management Mode (SMM) memory. " [45] The vulnerability affects the security of data in the SGX enclave as well as the attestation protocol. Intel has published microcode and firmware updates to mitigate this attack.
- 2022: "ÆPIC Leak enables attacks against SGX enclaves on Ice Lake CPUs, forcing specific data into caches and leaking targeted secrets, [...] We show attacks that allow leaking data held in memory and registers. We demonstrate how ÆPIC Leak completely breaks the guarantees provided by SGX, deterministically leaking AES secret keys, RSA private keys, and extracting the SGX sealing key for remote attestation." [46] Intel will try to mitigate this attack by microcode, SDK, and firmware updates. It is unclear at this time, however, how effective these mitigations will be. Since a TCB recovery is planned only for April 2023, systems are currently vulnerable.

As this (incomplete) list of attacks show, a modern CPU has many problematic interdependencies between its execution units (implemented primarily to speed up operation) that can lead to exploitable bugs, especially in conjunction with a complex technology like SGX. As several possible attacks have been discovered, there can be no assurance that the implementation is bug-free at this moment. There might be zero day exploits available to bad actors undermining the security of the whole SGX ecosystem.

Intel TDX

Intel Trust Domain Extensions (TDX) [47] introduces new functionality to the Intel processor line (starting with the Sapphire Rapids server processors).

Overview

TDX enable the deployment of hardware-isolated VMs called Trust Domains (TDs). These TD VMs are isolated from non-TD software on the host platform, including the VMM and hypervisor. By doing so, Intel TDX provides a means to enhance confidential computing, safeguarding TDs from various software attacks and helping to reduce the TD Trusted Computing Base (TCB). The technology empowers cloud tenants to exert greater control over their data security and intellectual property protection, while also enabling Cloud-Service Providers (CSPs) to offer managed cloud services without compromising tenant data to malicious actors.

Intel TDX leverages Intel Virtual Machine Extensions, instruction set extensions, memory-encryption technology, and a CPU-attested software module to create its solution. Through these technologies, Intel TDX provides several capabilities to Trust Domains. For example, it offers memory and CPU state confidentiality and integrity to protect sensitive IP and workload data from most software-based and many hardware-based attacks. Additionally, Intel TDX empowers workloads to exclude firmware, software, devices, and operators of the cloud platform from the trusted computing base, thus promoting more secure access to CPU instructions, security, debug, and other technologies. Figure 38 gives an overview of the platform components and functions that are part of the TDX TCB and those that lie outside of the TCB and therefore need not to be trusted.
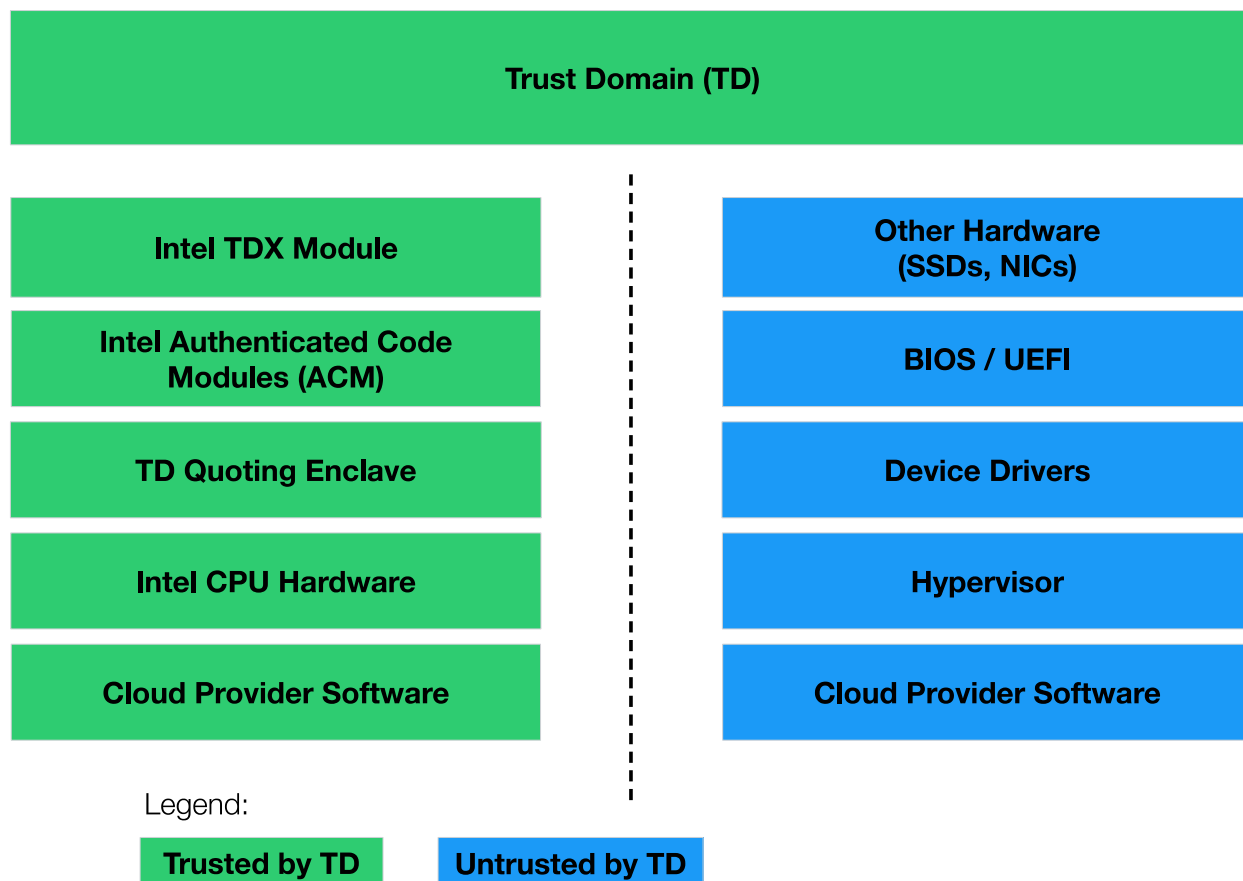
Trust Domain (TD)	
Intel TDX Module	**Other Hardware (SSDs, NICs)**
Intel Authenticated Code Modules (ACM)	**BIOS / UEFI**
TD Quoting Enclave	**Device Drivers**
Intel CPU Hardware	**Hypervisor**
Cloud Provider Software	**Cloud Provider Software**

Legend:

Trusted by TD	**Untrusted by TD**

FIGURE 38: TRUST BOUNDARIES FOR TDX

Remote attestation is also available to enable the owners and consumers of the service to digitally verify the version of the TCB and ensure that the workload is running on an Intel-TDX-enabled platform within a TD before providing workload data. Furthermore, Intel TDX enhances defence against limited forms of attacks that use physical access to the platform memory, such as offline, dynamic-random-access memory (DRAM) analysis, and active attacks of DRAM interfaces, including capturing, modifying, relocating, splicing, and aliasing memory contents.

TDX relies on previous security functions built into Intel processors such as TXT (Trusted Execution Technology). TDX entry points are defined as TXT ACMs (authenticated code modules) to enter a new mode of the CPU called Secure-Arbitration Mode (SEAM). SGX enclaves are not supported in TDs, however.

As the protection guarantees apply on the VM level, Intel TDX is very similar to the earlier approach by AMD with their AMD SEV architecture (compare Section 0).
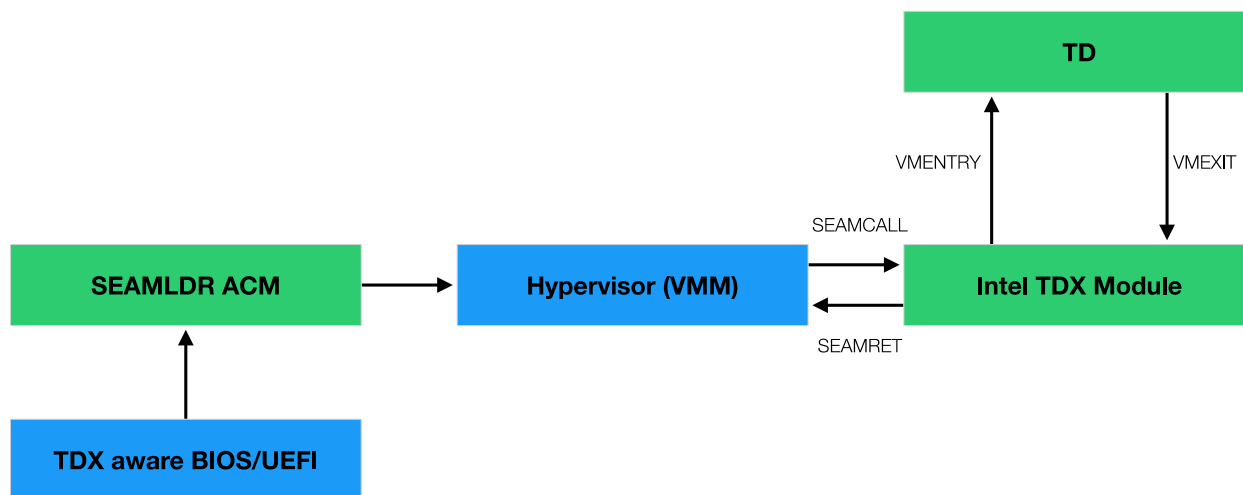
FIGURE 39: INTEL TDX SYSTEM BOOT FLOW

A TDX-aware system BIOS must initialise the system for TDX, allocating SEAM-protected memory and using a SEAMLDR ACM to put the TDX module into SEAM mode, passing then control to the hypervisor or VMM (Figure 39). The Intel TDX module acts as a gateway between the VMM and the TD and helps to securely initialise and manage the TDs through use of SEAMCALL and SEAMRET calls [48].
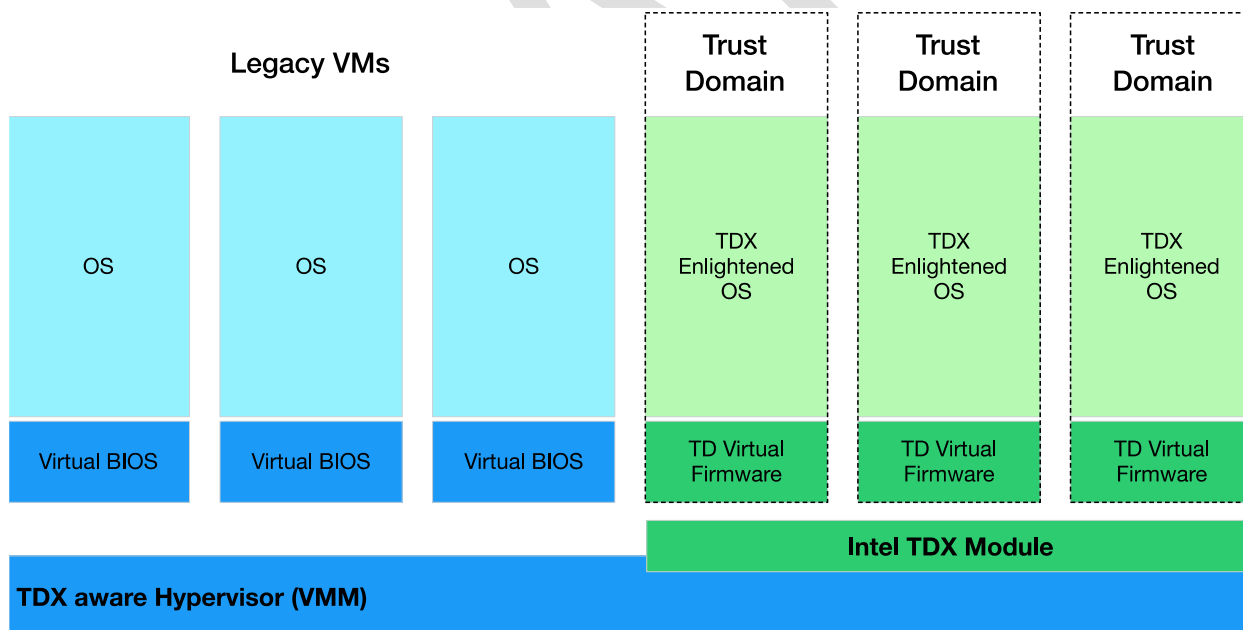

FIGURE 40: INTEL TD VIRTUAL FIRMWARE

In contrast to regular, unprotected VMS, TDs require use of a special TD virtual firmware that is called from the TDX module to protect the chain of trust up until starting the "TDX Enlightened OS" (Figure 40). This term means that the operating system is aware of running in a TD and exposes specific behaviour as described in the Intel TDX specifications. TDX guest support has been added to the Linux Kernel version 6.2 [49].

Since in Intel TDX the VMM is not part of the TCB, the TD virtual firmware must also check all VMMs calls and act as a secure gatekeeper between untrusted VMM and trusted TDX Enlightened OS [50].

Practical Experiences

We currently do not have access to a Sapphire Rapids system with Intel TDX, and there are currently no offerings in the public cloud.

The practical insights gained from comparing it to AMD SEV-SNP would be rather limited, however, as both technologies offer full protection and remote attestation functionality for an entire VM. Therefore, for practical purposes in the project, we will rely on Intel SGX and AMD SEV as two alternative implementation approaches.

Known Attacks

Since Intel TDX has only been available for a brief time with the Sapphire Rapids family of server processors, no known attacks have been published at the time of writing.

AMD SEV-SNP

AMD SEV is AMD's approach to mitigate the risks of outsourcing the processing of enterprise data into the cloud.

Overview

The technical infrastructure that forms the cloud is owned by the cloud provider and thus under his full control. This includes the server hardware, as well as the software components that allow the colocation of multiple virtual machines on a single host.

Like Intel SGX, the AMD SEV approach reduces the TCB to the AMD hardware and firmware as well as the SEV-SNP-protected VM (see Figure 41).
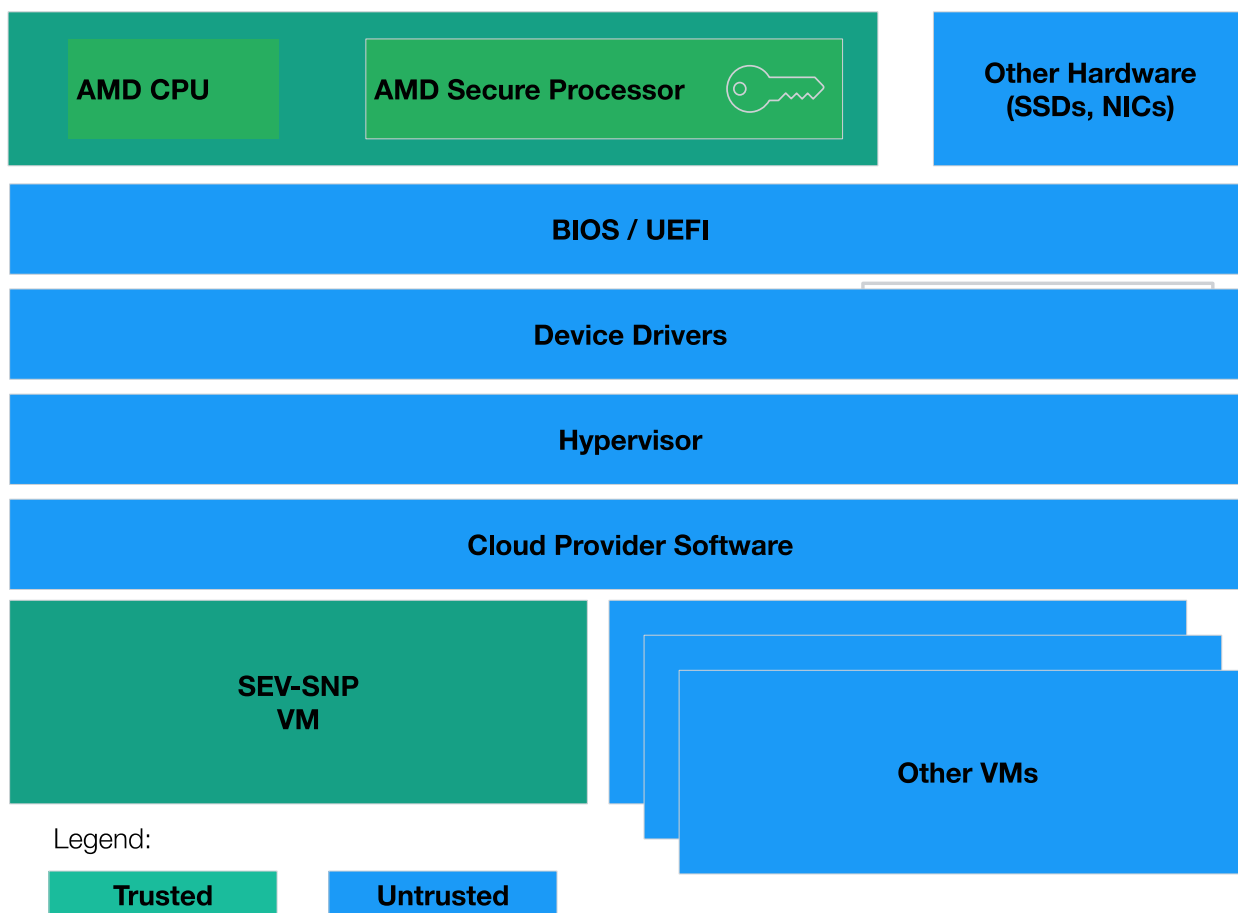
FIGURE 41: HIGH-LEVEL OVERVIEW OF AMD SEV-SNP ARCHITECTURE AND TRUST MODEL

Security concerns impede the deployment of confidential data and applications in cloud scenarios. The potential threats range from misconfiguration of software components over cloud provider admin access to foreign government access.

To counter these threats, AMD developed Secure Encrypted Virtualization technology (SEV). SEV's goals are twofold:

1. Prove the correct deployment of virtual machines.
2. Offer virtual machine protection at runtime.

To achieve the first goal, a dedicated co-processor, the Platform Security Processor (PSP), creates a cryptographic hash of the components that form the initial virtual machine state, similar to the remote attestation feature of a TPM.

The second goal is achieved by using memory encryption with virtual machine-specific encryption keys. These keys are generated upon creation of a virtual machine and are stored inside the PSP that uses its own private memory. This memory is not accessible from the main processor, which is executing software components controlled by the cloud provider such as the hypervisor. A hardware memory encryption unit provides transparent encryption and decryption using the virtual machine-specific key when the respective virtual machine is scheduled. Even though the hypervisor level is traditionally more privileged than the guest level, SEV separates these levels through cryptographic isolation.

The AMD attestation workflow is depicted in Figure 42: To allow the authentication of an SEV platform, each SEV-enabled platform contains a key pair that is unique to this platform. The public

key is signed by AMD and given a platform-specific ID; a guest owner can obtain this key from an AMD key server. This key is the basis for a remote attestation protocol that enables the user to verify the correct deployment of his VM, including that SEV protection is in place. Only then he will inject a guest secret, e.g., a disk encryption key, into the guest VM using a secure channel between the PSP and the guest owner. This remote attestation feature is a key component when using the infrastructure of an untrusted cloud provider. It provides cryptographic proof that the cloud provider uses an authentic AMD platform with SEV enabled and deployed the virtual machine according to the owner's configuration.
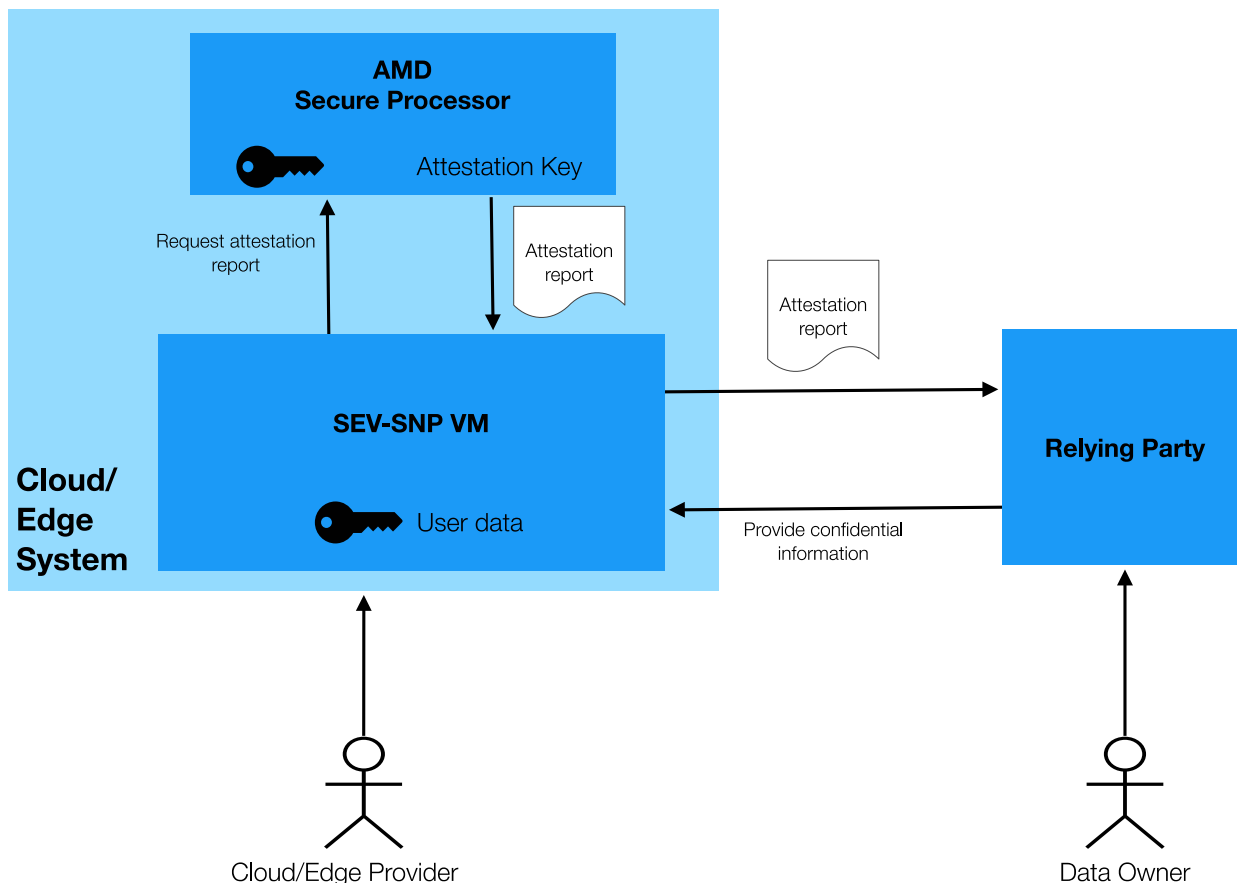

FIGURE 42: AMD REMOTE ATTESTATION

Practical Experiences

Refer to "A.1.2Practical Experiences with AMD-SEV" for details about our experimentation with AMD-SEV.

Known Attacks

Many of the successful attacks on SEV discovered by researchers in the last years have been addressed by the introduction of Secure Nested Pages (SEV-SNP) in the latest incarnations of AMD EPYC processors based on the Zen 3 architecture. Zen 1 and Zen2-based products remain susceptible to attacks that undermine the whole security of the SEV architecture – these systems should not be used for sensitive workloads anymore.

For Zen 3, the only currently known attack vector identified by Robert Buhren [51]in his PhD thesis (developed at the chair for Security in Telecommunications (SECT) at Technical University Berlin, an affiliated institute of Deutsche Telekom Innovation Laboratories) is a power-glitching attack

that requires physical manipulation of the EPYC CPU in order to disrupt the signature checking of the PSP firmware, therefore allowing the loading of a manipulated firmware that subverts the security of the SEV-SNP implementation. This is a less scalable attack than the software attacks on Intel SGX.

Intel vs. AMD Evaluation

The AMD and Intel approach differ significantly when it comes to the actual work needed for developers to package their applications for secure cloud computing.

The Intel SGX technology needs a significant programming effort to split existing applications into secure (SGX) and non-secure parts. Additionally, new code must be developed and audited to securely transfer data between these two parts.

In contrast to that, AMD just offers to package a whole VM into a secure container with the target application running on top of a regular operating system in the container.

Table 2 shows a breakdown of the difference as well advantages and disadvantages of the approaches.

TABLE 2 AMD VS. INTEL EVALUATION

	AMD	Intel SGX	Intel TDX
Ease of integration	Easy: just package the application in a standard VM	Difficult: break down application in secure and non-secure parts	Easy: just package the application in a standard VM
Trusted Computing Base	Large: A whole operating system and the whole application.	Small: If designed correctly, only the part of the application in the SGX container needs to be trusted.	Large: A whole operating system and the whole application.
Architectural approach	SEV relies on the firmware of the AMD Secure Processor.	SGX relies on microcode in the processor.	TDX relies on microcode in the processor.
Remote attestation	Involves contact to AMD key server to download attestation credentials need to verify attestation reports (can be cached).	Involves contractual agreement with Intel and contact to Intel SGX Provisioning Certification Service to load attestation credentials, local caching possible with DCAP.	Involves contact to Intel key server to download attestation credentials need to verify attestation reports (can be cached).
Encryption	Memory of SEV-protected VMs is encrypted.	Memory of SGX enclaves is encrypted.	Memory of TDX-protected VMs (TDs) is encrypted.
Known attacks	There is only one currently known weakness which is a physical attack that requires access to and manipulation of CPU on the hardware level.	There are currently known attacks that call into question the security of the SGX platform. It is unclear if the available microcode updates by Intel resolve these problems.	There are no known weaknesses at the moment because the platform is quite new.
Licensing	No licensing requirements at the time of writing.	Using the Intel infrastructure for production use, especially the Intel SGX Provisioning Certification Service, requires a license by Intel. Intel controls access to SGX protections through this licensing scheme.	No licensing requirements at the time of writing.

The findings can be summarised as follows: AMDs approach is less complex and easier to implement, since it basically allows to package a whole guest OS into a secure VM. Also, AMDs attestation process is simpler and is not connected with any licensing issues. Intel now implements a similar approach with Intel TDX.

The Intel approach of using microcode allows for easy security updates but needs for developers to make changes to applications to adapt to SGX. Additionally, Intel requires licensing for using SGX in production mode.

The known attack to both technologies question the security guarantees offered by their vendors in question. Most probably, these will be incrementally addressed by new CPU hardware releases or microcode/firmware updates. The attacks against AMD SEV-SNP require physical manipulation while the against Intel SGX can be executed in software without physical access.

3.5.2.3 Software frameworks

As has been identified in the previous section, developing software for secure cloud computing, specifically for the Intel SGX technology, requires engagement of the developers. Therefore, we will look at some commercial and open-source frameworks building on the hardware solutions introduced earlier to ease and simplify the workload for DevOps teams.

SCONE

Using the SCONE platform, data and code can be protected not only while in transit (i.e., on the network), but also at rest and in use. SCONE is currently available for the Intel SGX platform only. SCONE protects the confidentiality and integrity of the data and the code without needing to modify or recompile the application, which would otherwise be necessary when developing for SGX [52].

One new aspect of confidential computing is that applications can be protected against privileged software like the operating system and the hypervisor. To certify that the data and code of a native application are adequately protected, users need to ensure that all software and hardware components are sufficiently protected. SCONE isolates each service individually – this reduces not only the attack's surface but also the effort to certify that an application is properly protected. The isolation enables the outsourcing of the management of components like the operating system and Kubernetes to a cloud or service provider. SCONE decouples the integrity and confidentiality of data and code from all other software components and from the entity that manages the software and hardware stack.
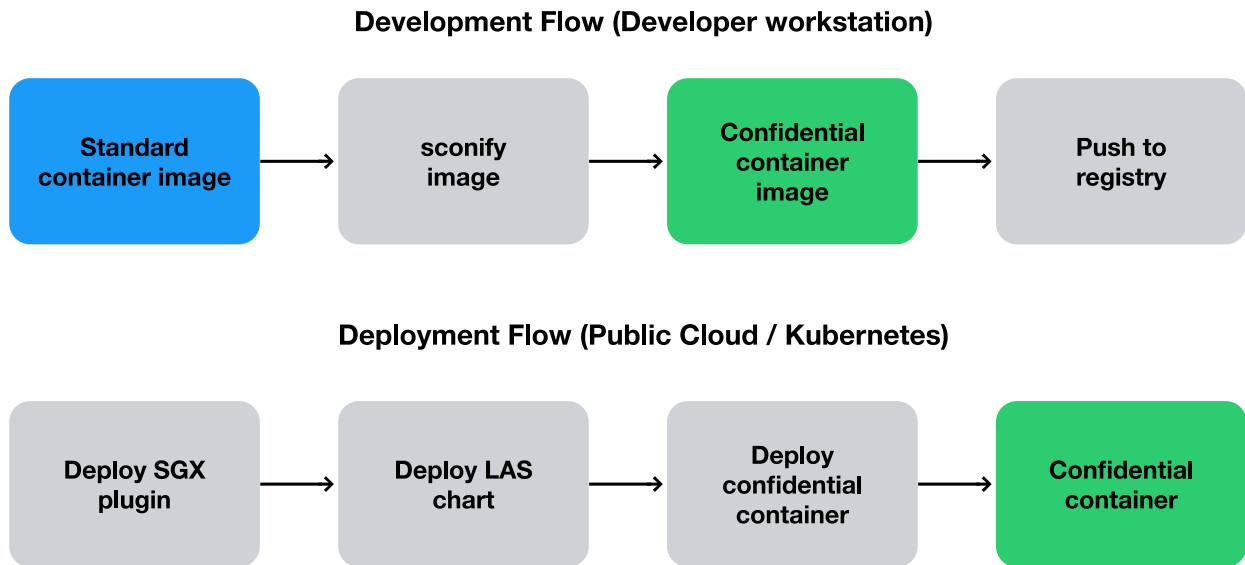
Development Flow (Developer workstation)

Deployment Flow (Public Cloud / Kubernetes)

FIGURE 43: SCONE-BASED DEVELOPMENT AND DEPLOYMENT WORKFLOW
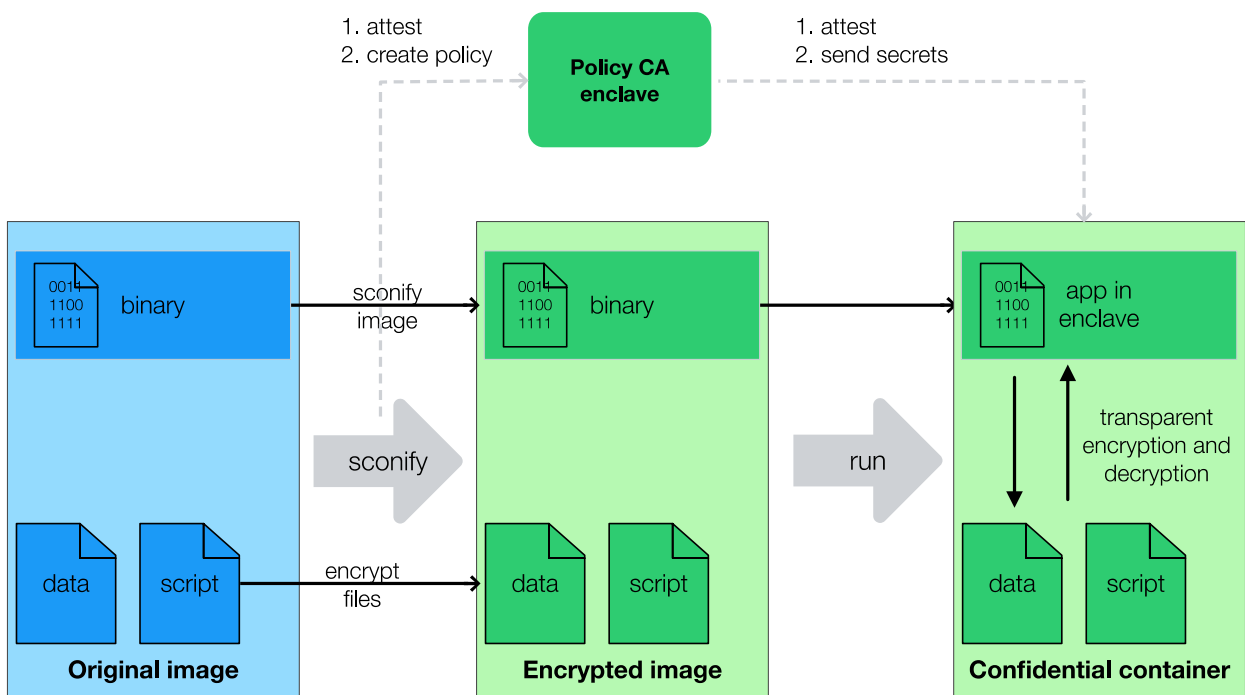

FIGURE 44: DETAILED SCONE WORKFLOW

Using SCONE, it is possible to transform ("sconify") an existing application into a confidential application without any need to modify the application. This process can be integrated into existing development and deployment workflows.

An application typically consists of one or more containers where one container hosts one service. A container is deployed with the help of a container image as part of a CI/CD pipeline. This container image can be transformed into a confidential container image with the help of a

special container image called `sconify_image`. There is no need for an SGX-capable environment for this process.

In a Kubernetes context, where confidential container images are deployed on a Kubernetes Cluster, a confidential container image is deployed in the same way as a native Kubernetes image: via `kubectl` or via `helm`. Scone, however, requires some extra services: the SCONE SGX Plugin and a local attestation service on each node. From the point of view of Kubernetes, a confidential container is handled in the same way as a native container.

For running services in production mode, services would be set up in a production Kubernetes cluster first. In a second step, deployment and update of confidential services can be handled automatically.

A production Kubernetes cluster, in addition to the SCONE SGX Plugin and a local attestation service, also needs a production-mode Configuration and Attestation Service (CAS). Installation of these services can be facilitated by helm charts. The encrypted CAS database must be stored in a persistent Kubernetes volume. CAS can be deployed in a primary or backup system configuration: When the current primary fails, a backup CAS will be started and will take over the CAS service automatically.

Gramine

Overview

Originally, Gramine was designed as a lightweight guest OS to make it easier to run applications designed for one system to work on others. Gramine does this by moving APIs from the operating system (OS) kernel into a user space library [53]. Gramine uses a platform adaptation layer (PAL) that can easily be adapted to new host systems. If the PAL interface can be implemented on a system, it can run Linux applications. Because Gramine is implemented as a user space library (comparable to a unikernel) Gramine is much more light weight than, e.g., a virtual machine (VM).

A particular use case for Gramine is to support running Linux applications using the Intel SGX (Software Guard Extensions), which is called Gramine-SGX. With Intel SGX, applications are secured in hardware-encrypted memory regions (called SGX enclaves). SGX protects code and data in the enclave against privileged software attacks and against physical attacks on the hardware off the CPU package (e.g., cold-boot attacks on RAM). Gramine is able to run unmodified applications out-of-the-box inside SGX enclaves, without any effort on part of the developers to port the application to SGX.
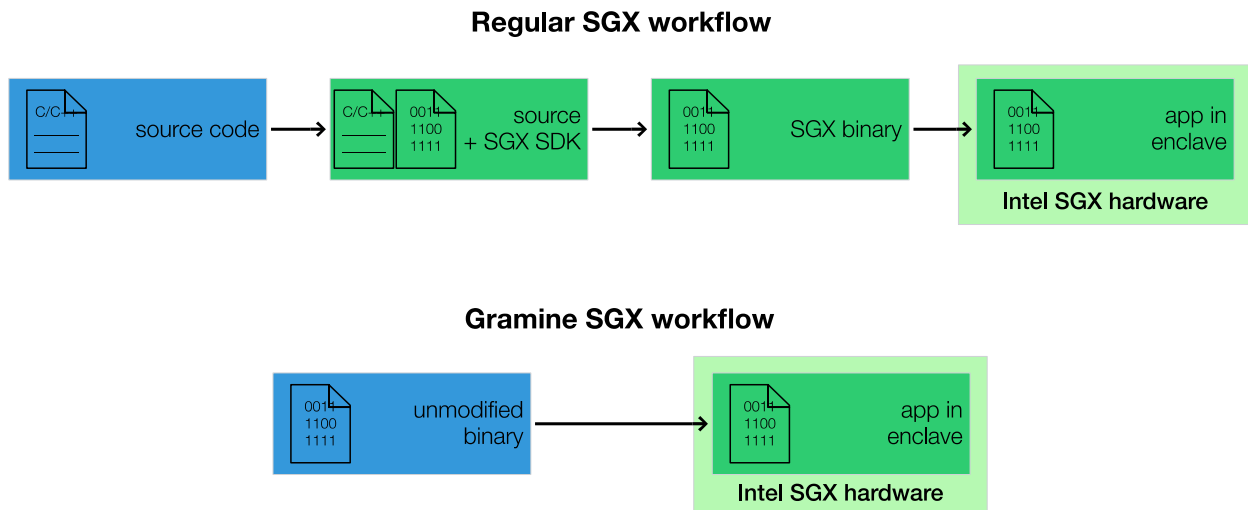
Regular SGX workflow

Gramine SGX workflow

FIGURE 45: GRAMINE-BASED DEVELOPMENT WORKFLOW

Figure 45Figure 45 shows both the regular and the Gramine-based workflow. Normally, the developer would have to port an application to SGX by splitting it into the "regular", unsecured part and the SGX enclave-based part. Gramine, however, changes the unmodified executable such that it runs in the Gramine environment using the PAL with all the SGX security guarantees intact.

Practical Example

Refer to "A.1.3.1 Practical Example with Gramine" for details of our tests with the Gramine framework.

3.5.3 Trusted Execution Environments by Public Cloud Providers

This section provides an overview of the actual implementations of AMD SEV and Intel SGX available from three important public cloud providers, namely Microsoft, Amazon, and Google.

For each provider, we strive to document the results of initial testing of the security mechanisms. These will be extended in later deliverables of the project to give a more complete picture of the security guarantees that can be expected from each of the provider's implementations.

3.5.3.1 Microsoft Azure

Overview

Microsoft offers AMD- and Intel-based VMs in the Azure Cloud. While Intel SGX-based confidential computing has been part of the Azure portfolio for some time, AMD SEV-SNP-based VMs are quite new and have just left preview status to enter general availability at the time of writing.

Microsoft offers a dedicated remote attestation framework called Microsoft Azure Attestation that relies on a Microsoft-provided SGX enclave (the Azure Attestation TEE) that can provide attestation reports for Intel SGX enclaves, AMD SEV-protected confidential VMs as well as TPM- or Trusted Launch-protected resources in the Azure cloud [54]. The trustworthiness of this Microsoft-supplied attestation technology as well as the possibility to implement SGX or

SEV attestation without using Microsoft Azure services will be studied as part of the demonstrators implemented in the project going forward.

AMD SEV-SNP

Azure confidential VMs undergo attestation as part of their boot phase. This process is opaque to the user and takes place in the cloud operating system in conjunction with the Microsoft Azure Attestation and Azure Key Vault services. Confidential VMs also allow users to perform independent attestation for their confidential VMs. This attestation happens using new tooling called Azure confidential VM Guest Attestation. Guest attestation allows customers to attest that their confidential VMs are running on AMD processors with SEV-SNP enabled. AMD SEV-SNP is supported by DCa machine types in the Azure cloud [55].

Intel SGX

SGX does not protect the whole VM like AMD SEV but requires [33]the developer to setup an SGX enclave specific for an application that runs on an otherwise unprotected operating system. SGX does, however, require some CPU and system hardware support as well as some SGX-specific drivers and other software. Therefore, Intel SGX VMs runs only on specialized hardware in specific regions and on DCsv2-series or DCsv3/DCdsv3-series instances in these regions (at the time of writing).

Practical Experience

Refer to "A 1.4.1 Microsoft Azure" for details on our test with Intel SGX and AMD-SEV on Microsoft Azure.

Summary and Further Work

We will need to test the Azure guest attestation feature to independently check the integrity of AMD SEV instances. Currently, there is only work-in-progress GitHub repository with not much documentation [56]. At the moment, remote attestation relies on automatic processes in the Azure cloud and is opaque to the cloud user.

3.5.3.2   Google Cloud

Overview

Google calls its approach to cloud security "Confidential Computing" [57]. Confidential Computing is the protection of data in-use with hardware-based Trusted Execution Environment (TEE). TEEs are secure and isolated environments that prevent unauthorized access or modification of applications and data while they are in use. End-to-end encryption is comprised of three states.

- Encryption-at-rest protects user data while it is being stored.
- Encryption-in-transit protects user data when it is moving between two points.
- Encryption-in-use protects user data while it is being processed.

Confidential Computing provides the last piece of end-to-end encryption: encryption-in-use.

To this end, Google introduces the term "Confidential VM." A Confidential VM is a type of Compute Engine VM that ensures that data and applications stay private and encrypted even while in use.

Confidential VM runs on hosts with AMD EPYC processors, which feature AMD Secure Encrypted Virtualization (SEV), offering the following security benefits:

1   Isolation: Encryption keys are generated by the AMD Secure Processor (SP) during VM creation and reside solely within the AMD System-On-Chip (SOC). These keys are not accessible by Google, offering improved isolation.

2   Attestation: Confidential VMs use Virtual Trusted Platform Module (vTPM) attestation. Every time an AMD SEV-based Confidential VM boots, a launch attestation report event is generated that can be checked by remote parties before transferring confidential data.

3   High performance: Enabling Confidential VM has little or no impact on most workloads, with only a 0-6% degradation in performance, since it relies on protections built into the AMD server platform anyway.

At its NEXT 2022 conference Google introduced a new product: Confidential Space. Built on Confidential Computing, and leveraging remote attestation, Confidential Space runs workloads in a Trusted Execution Environment (TEE). Together with the hardened version of Container-Optimized OS (COS), data contributors can have control over how their data is used and which workloads are authorized to act on it. Finally, Confidential Space blocks the workload operator from influencing the workload in any way. At the time of writing, it is not clear how Confidential Space is differentiated from Confidential Computing alone. The announcement can, however, at least be interpreted as a marketing push, further emphasizing the secure computing angle for Google Cloud.

Practical Experience
Refer to "*A.1.4.2 Google Cloud*" for details on our tests with Google Cloud.

3.5.3.2.1   Summary and Further Work

Google makes it easy to get started with confidential computing. With the default settings, however, the cloud user still must rely very much on Google-provided VMs and services. Therefore, one still must trust Google to do the right thing.

We will investigate further how to import own VM images into the Google cloud and run remote attestation for VMs instantiated from these images in a secure way, excluding Google from the trusted computing base (TCB).

3.5.3.3   Amazon Web Service

Overview

Amazon Web Service does not offer hardware-based technological safeguards for server processors such as AMD SEV or Intel SGX. Instead, Amazon relies on a number of hardware and software features under the "Nitro" brand, e.g., the Nitro Security Chip, NitroTPM and proprietary components *[58]*.

AWS Nitro allows the user to create isolated environments that are strongly protected from other parts of the environment by means of the abovementioned hardware features of the physical cloud infrastructure. This allows a user to create or obtain enclave-based applications that they trust to operate on sensitive data or embody secret proprietary algorithms, without having to trust the security of their operating system, cloud administrators, or bad actors that gain access to their compute infrastructure. These enclaves provide no persistent storage,

external networking, or operator-based access; they can only communicate through a trusted channel to the instance that created the enclave.

AWS Nitro Enclaves use a secure virtual socket features present in the Linux kernel for some time, as the only communication channel between the "trusted" software running within the enclave and the "normal" or "untrusted" software running in the EC2 instance. The approach is therefore similar to Intel SGX, but without any specific CPU support, since AWS Nitro Enclaves are processor agnostic and work with most Intel and AMD-based Amazon EC2 instance types.

An important component of Amazon's approach on the software side is the Nitro Hypervisor, which is a firmware-based hypervisor that is responsible for using processor hardware features to strongly isolate physical system resources in creating EC2 instances and AWS Nitro Enclaves. The Nitro Hypervisor is purpose-built to meet the security and operational needs of AWS and does not rely on industry standards or open-source virtualization environments.

Summary and Further Work

Since Amazon uses proprietary hardware and software to secure its Nitro enclave offering, the trusted computing base of the solution is fully under Amazon's control. Because of this, it is unclear how it can be guaranteed that Amazon as cloud provider does not have access to confidential data in the Nitro-protected enclaves after all. The solutions offered by Microsoft and Google, however, are based on the separation of concerns between owner of the computing infrastructure and manufacturer of the CPU hardware (i.e., AMD and Intel). A successful breach of the security guarantees would require collusion between cloud provider and CPU manufacturer in the case of Microsoft and Google.

Because of these considerations AWS Nitro is deemed to be out-of-scope for this document and will not be analysed further during the project.

3.5.4 Certificate Provisioning for the cloud

In the previous sections, we have laid the groundwork for securely deploying SPIRIT workloads in public or private cloud or even edge resource (e.g., on the factory floor). This protects data-in-use and data-in-transit.

In this section, we show how to protect data-in-transit using certificates from DT-Sec Telekom trust centre and combine this industry-standard approach into a novel way with the security guarantees offered by Confidential Computing.

3.5.4.1 Overview of ACME

We propose a certificate management solution for SPIRIT based on the ACME protocol. ACME is a certificate management protocol defined in RFC 8555. A popular use of ACME is to request certificate from Let's Encrypt, a free public CA. The protocol is also supported by some commercial CAs.

ACME builds on several existing technologies, most of which are commonly used in web API designs:

- HTTPS (transport)
- JSON (message format)
- TLS server authentication (authentication of messages from server)

- JWS (authentication of messages from clients)
- PKCS #10 (CSR)

ACME supports several certificate management use cases:

1. requisition (a.k.a. enrollment)
2. certificate revocation
3. subject identifier validation

In ACME, a client needs to have an account to be able to use the service of an ACME server. A client account is identified with an account key pair. Among certificate enrolment protocols, ACME is unusual in that it has provisions for extensible, automated in-protocol processes for validating identifiers (e.g., DNS domain) of certificate subjects. This allows ACME accounts to be set up by clients without pre-established credentials, as a client's eligibility to receive a certificate with a particular subject identifier not based on trust of the client, but independent validation of the client's control of the subject identifier. RFC 8738 extends ACME to issue certificates with IP address identifiers using http-01 and tls-alpn-01 challenges.

ACME has mechanisms for integrating with external systems. "External account binding" allows an ACME account to be tied to an account external (using a MAC-authenticated credential) and allows authorization to be performed via the external account. ACME also support an operation called preauthorization, which allows a client to request authorization for a subject identifier outside of the context of a certificate request sequence.

In ACME, a certificate request sequence involves the creation and update of several types of objects:

- order
- authorization
- challenge

Messages from the client are authenticated using JWS signed with the client's account private key. To protect against replay attacks, requests from the client contain a replay nonce. Some of the request-response exchanges are asynchronous in the general case. In those exchanges, after receiving an immediate response from the server, the client may need to poll the server periodically until a status change has occurred.

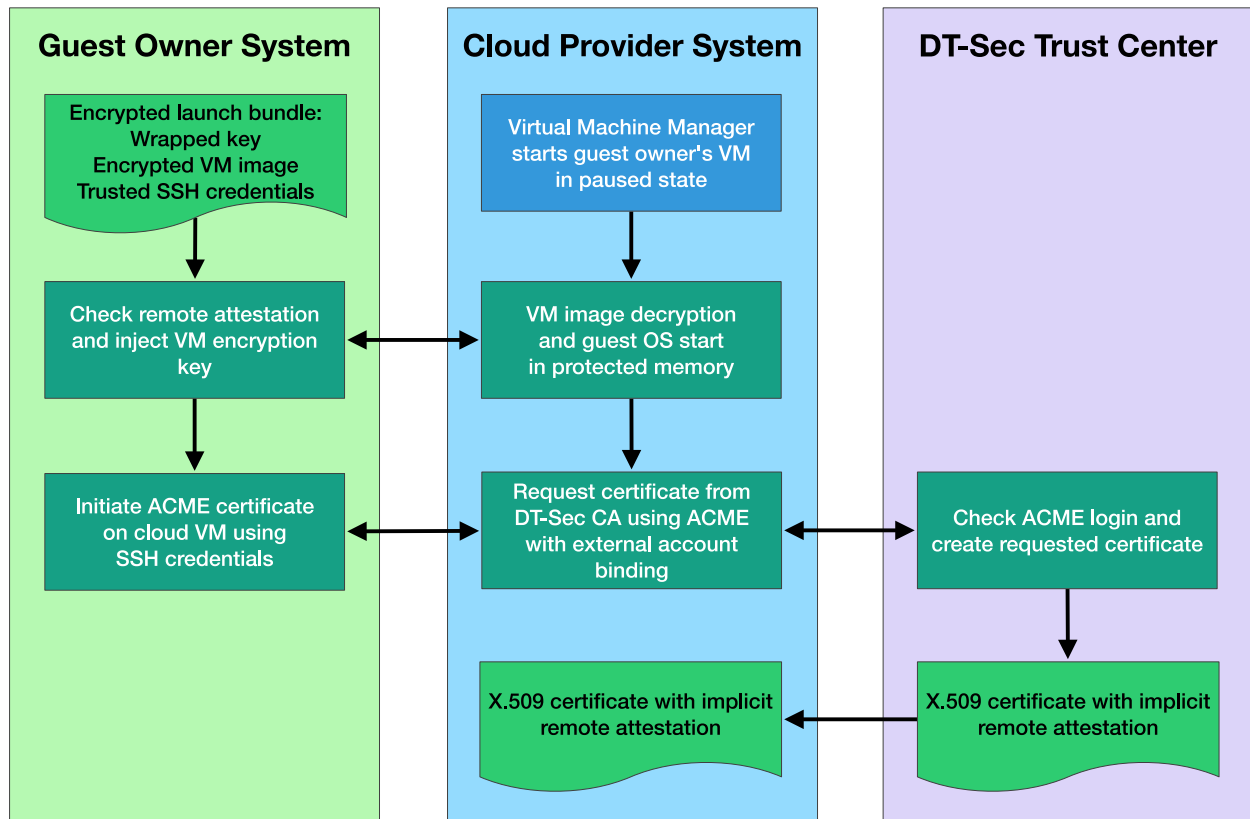### 3.5.4.2 Certificate Management based on Confidential Computing

FIGURE 46: CERTIFICATE PROVISIONING FOR THE CLOUD USING CONFIDENTIAL COMPUTING.

Figure 46 describes a scenario involving the deployment and management of a trusted virtual machine (VM) in order to obtain certificates from a trusted authority without requiring remote attestation by the communication partners. The process will be implemented as follows:

1. Precondition: Deploy encrypted VM with embedded SSH key
   In this step, a virtual machine is deployed with its disk encrypted. The VM also contains an embedded SSH key, which will be used for secure remote access and management. By encrypting the VM's disk, the data stored within it is protected from unauthorized access in case of theft or unauthorized access to the underlying infrastructure.

2. Passes Remote attestation, send encrypted disk key
   Remote attestation is a process that verifies the integrity and security of a remote system. In this case, the deployed VM undergoes remote attestation to ensure that it meets the required security standards and hasn't been tampered with. Once the VM successfully passes the attestation process, the encrypted disk key is sent to the VM. This encrypted disk key is used to access and unlock the encrypted disk on the VM.

3. Use SSH-based management to now trusted VM
   After the encrypted disk key is sent to the VM, SSH-based management is used to establish a secure remote connection with the trusted VM. SSH (Secure Shell) provides a secure channel for communication and remote administration. By using SSH-based

management, the administrator can securely interact with the trusted VM, perform various tasks, and manage its operations.

4. Start ACME client to request a certificate from DT-Sec trust centre
   Within the trusted VM, an ACME (Automated Certificate Management Environment) client is initiated. ACME is a protocol used for automating the issuance and management of digital certificates. The trusted VM uses the ACME client to communicate with the DT-Sec (Digital Trust Security) trust centre, a trusted authority responsible for issuing certificates. The ACME client generates a certificate signing request (CSR) and sends it to the trust centre, requesting a certificate for the VM.

Result: Only trusted VMs receive certificates. Communication partners do not need to perform remote attestation themselves.

As a result of the previous steps, the trusted VM receives the certificate from the DT-Sec trust centre. This certificate verifies the identity and authenticity of the trusted VM. The communication partners who want to establish secure connections with the trusted VM no longer need to perform remote attestation themselves. They can simply trust the certificate issued by the trusted authority, eliminating the need for repetitive attestation processes. This simplifies the communication setup and ensures that only trusted VMs receive certificates, enhancing security and trust in the system.

Overall, the described process involves deploying a secure and trusted VM, verifying its integrity through remote attestation, and obtaining certificates from a trusted authority. This allows secure communication with the VM without requiring each communication partner to perform remote attestation individually.

3.5.5 Intrusion Detection Approaches for SPIRIT Deployments

The attack surface of a distributed system – and a disaggregated SPIRIT deployment 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.
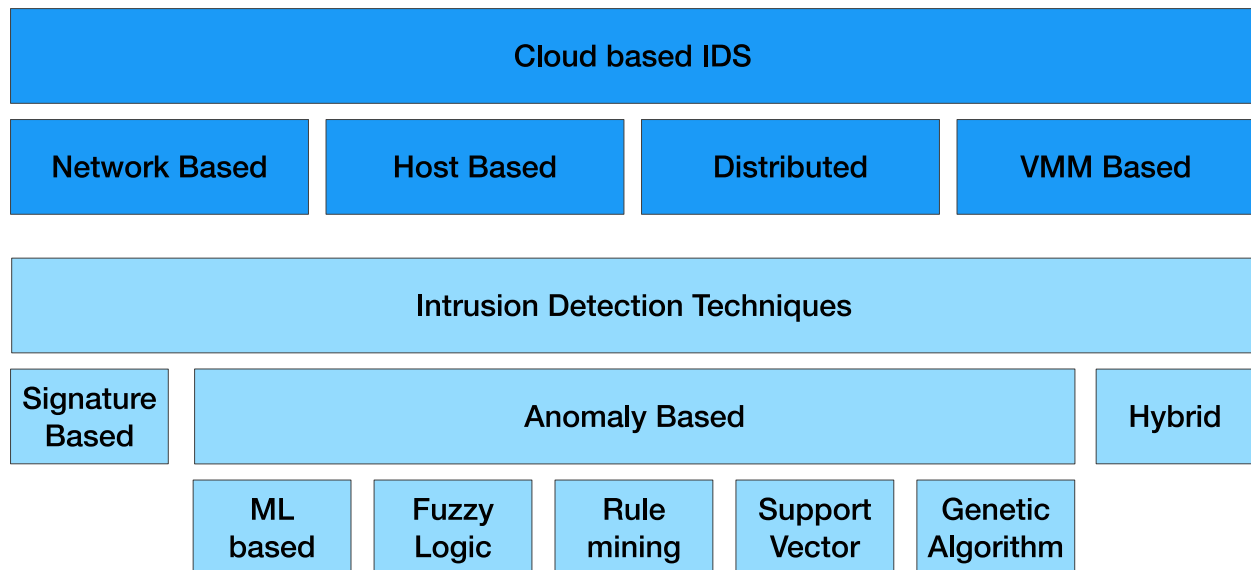
Additionally, distributed systems often involve communication between different components, which can involve multiple protocols and ports. This increases the potential for vulnerabilities and attack vectors that can be exploited by attackers.

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 centralized 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.

Cloud based IDS			
Network Based	Host Based	Distributed	VMM Based

Intrusion Detection Techniques						

Signature Based	Anomaly Based					Hybrid
	ML based	Fuzzy Logic	Rule mining	Support Vector	Genetic Algorithm	

Based on: Y. Mehmood et. al.: Intrusion Detection Systems in Cloud Computing: Challenges and Opportunities.
DOI: 10.1109/NCIA.2013.6725325, Corpus ID: 15265295.

FIGURE 47: OVERVIEW OF INTRUSION DETECTION APPROACHES

Intrusion detection systems (IDS, refer to Figure 47) play an important role in ensuring the security of distributed systems. These systems monitor network traffic and identify suspicious activity that may indicate an attempted intrusion or attack.

One approach to intrusion detection in distributed 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 distributed systems, and a combination of different intrusion detection techniques can be used to provide comprehensive security coverage.

3.5.6 Security Innovations for the SPIRIT Project

In the previous sections we have elaborated on the concept of end-to-end security with Confidential Computing at the centre. We have performed an analysis of state-of-the art, the offerings of public cloud providers, as well as some examples of existing open-source software.

Based on this conceptual work we have implemented a mechanism to create confidential test VMs (see Figure 48).
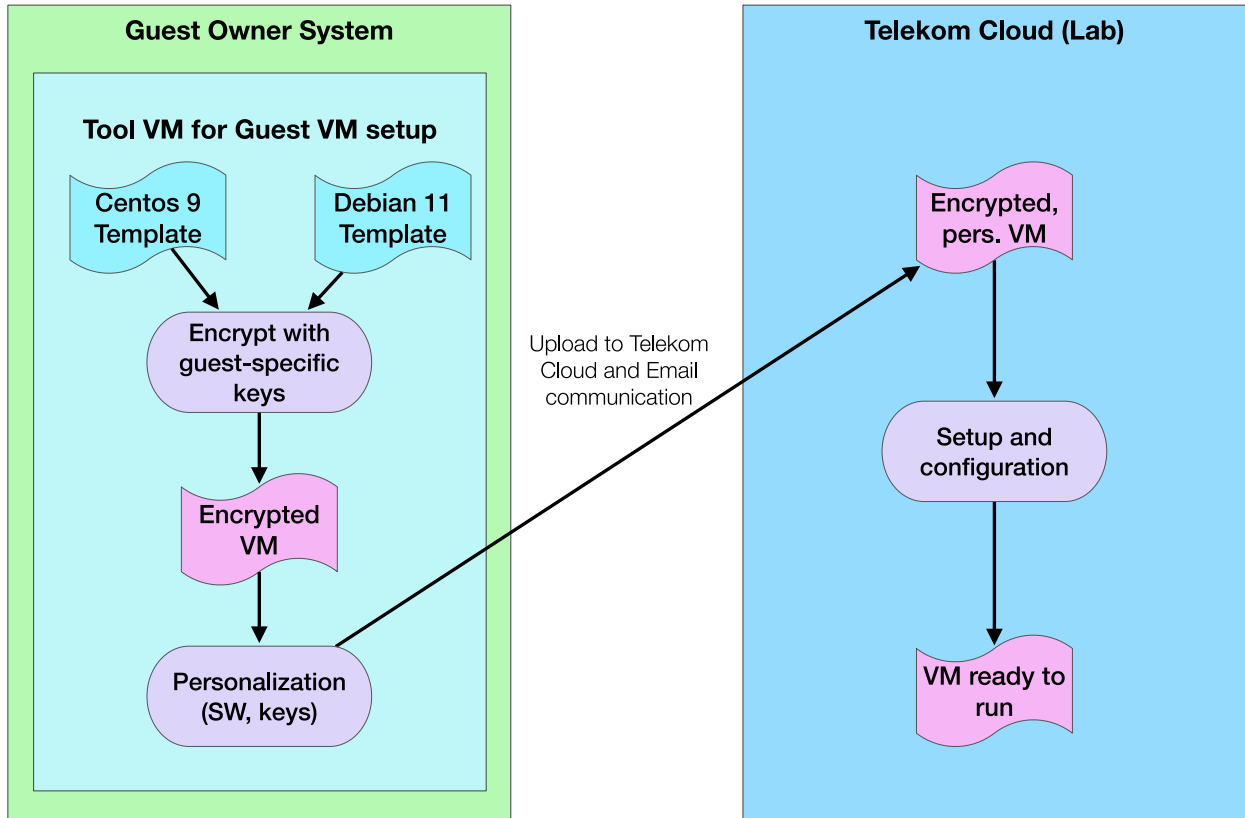
FIGURE 48: SETUP OF THE GUEST VM

Additionally, we have created a set of tools to deploy and manage these Confidential VMs (see Figure 49).
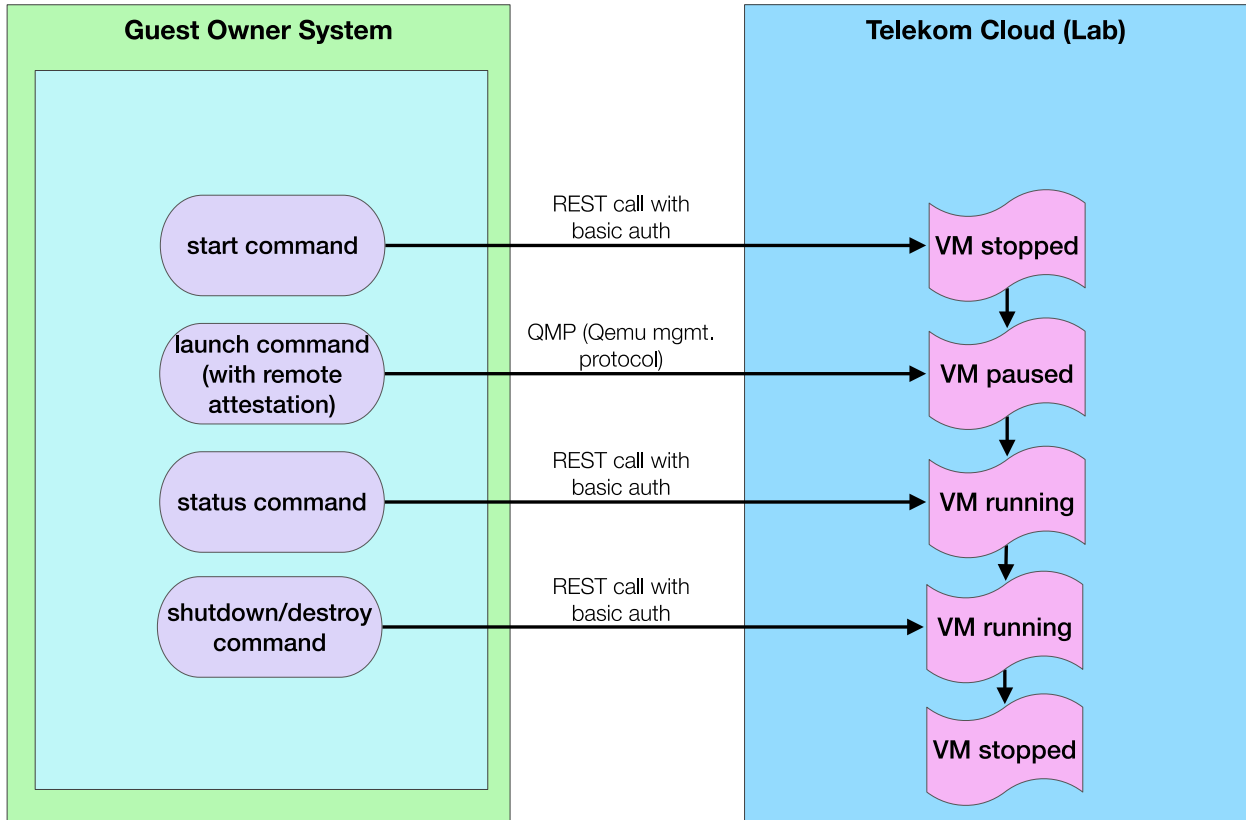
FIGURE 49: MANAGING THE GUEST VM

To allow for integration of and experimentation with these technologies we have set up a lab environment for project partners and Open Call partners to run confidential workloads in a cloud-like environment in our Deutsche Telekom Cloud Security Lab in Berlin (see Figure 50).
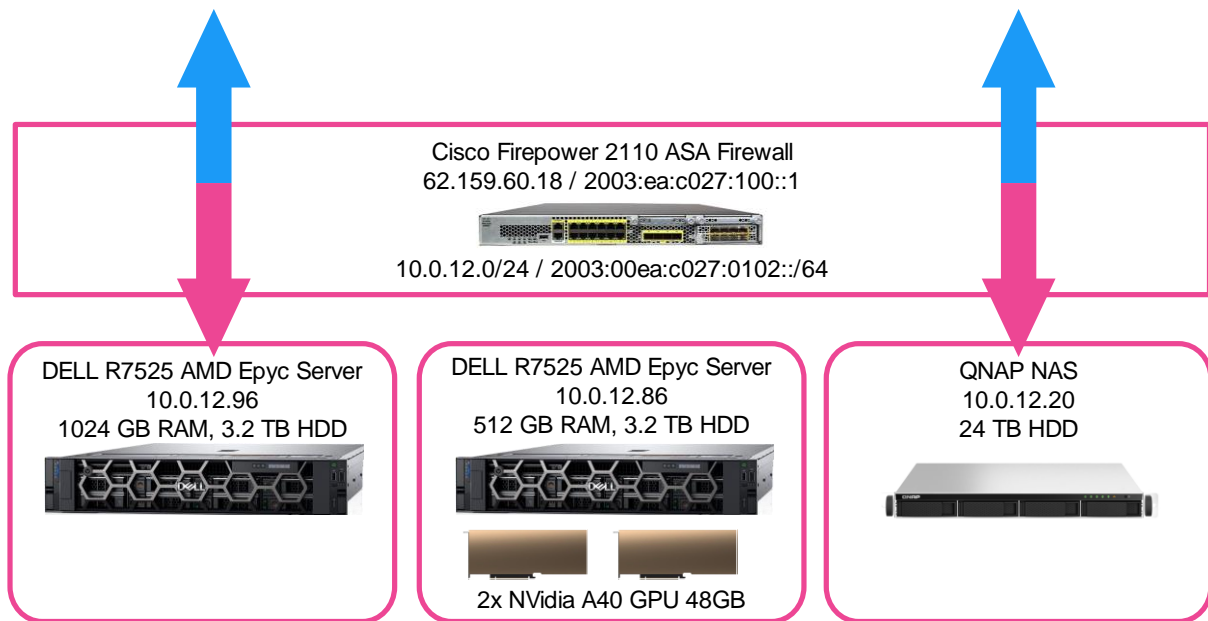

FIGURE 50: CLOUD SECURITY LAB INFRASTRUCTURE

The setup in our lab consists of two Confidential Computing-enabled servers, one of them with Nvidia-based GPU resources.

Technical details of these components are described in deliverable D4.2 [5].

Progress on the topics of certificate provisioning, intrusion detection, and identity management in the SPIRIT context will be covered in future versions of the deliverables 3 and 4. Interim documentation on certificate provisioning and intrusion detection is available to Open Call partners as needed.

3.6 SUMMARY

To summarise, the innovations presented are promising to improve the current immersive telepresence solutions found in application platforms and 5G network testbeds of the project partners. The integration of the innovation platform enablers into the test environments have been largely completed as described in D4.2 [5].

4   CONCLUSIONS

In the second version of the innovation platform enablers document, we presented an updated description of implemented immersive telepresence solutions, covering diverse aspects, such as content delivery, transport mechanisms and application platforms, and which are aligned with the evolved SPIRIT architecture.

In conclusion, we have implemented and tested platforms that include most of the immersive telepresence solutions presented in Sections 2 and that were integrated into the 5G network testbeds of the partners so that third parties can access them as part of the Open Calls for their experiments.

We have additionally provided new innovations on content, application, transport, network and security that align with evolved SPIRIT architecture and are being investigated as candidates for inclusion into the project testbeds.

The content innovations described in this report advance the human representation through enhanced video-driven avatar representations compared to using audio only. The ambition is to integrate into the testbeds and be experienced by the users.

The transport innovations aim to advance the technology by offering different transport mechanisms, in particular low latency DASH and WebRTC, and offer more use-cases and integration possibilities for open all partners.

The applications and network relevance through variance innovations aim to enhance the performance such bandwidth scale the current platforms to support multi-user conferencing use-cases.

The security innovations for different cloud deployment aim at enhancing the testbed functionalities and are applicable for different use-cases such as human-to-human communication and human-to-machine communications.

5 BIBLIOGRAPHY

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

[2] SPIRIT Consortium, "Use Case Requirements, System Architecture and Interface Definition (Second Version)," Project "Scalable Platform for Innovations on Real-time Immersive Telepresence", EC grant agreement 101070672, 2024.

[3] SPIRIT Consortium, "SPIRIT Platform (Second Version)," Project "Scalable Platform for Innovations on Real-time Immersive Telepresence", EC grant agreement 101070672, 2024.

[4] S. Moezzi, "Immersive Telepresence," *IEEE MultiMedia,* vol. 4, no. 1941-0166, p. 17, 1997.

[5] H. Fuchs, A. State and J. C. Bazin, "Immersive 3D Telepresence," *Computer,* vol. 47, pp. 46-52, 2014.

[6] A. Merlo, C. Sánchez Belenguer, E. Vendrell Vidal, F. Fantini and A. Aliperta, "3D model visualization enhancments in real-time game engines," *The international archives of the photogrammetry, remote sensing and spatial information sciences,* vol. 40, pp. 181-188, 2013.

[7] S. Vincke, R. de Lima Hernandez, M. Bassier and M. Vergauwen, "Immersive visualisation of construction site point cloud data, meshes and BIM models in a VR environment using a gaming engine," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences,* vol. 42, pp. 77-83, 2019.

[8] W. Paier, M. Kettern, A. Hilsmann and P. Eisert, "A Hybrid Approach for Facial Performance Analysis and Editing," *IEEE Transactions on Circuits and Systems for Video Technology,* 2017.

[9] W. Paier, A. Hilsmann and P. Eisert, "Unsupervised Learning of Style-Aware Facial Animation from Real Acting Performances," *Elsevier Graphical Models,* 2023.

[10] A. Baevski, H. Zhow, A. Mohamed and M. Auli, "Wav2Vec2: A Framework for Self-Supervised Learning of Speech Representations," *CoRR,* 2020.

[11] J. Thies, M. Elgharib, A. Tewari, C. Theobalt and M. Nießner, "Neural Voice Puppetry: Audio-driven Facial Reenactment," *ECCV,* 2020.

[12] [Online]. Available: http://tools.ietf.org/wg/rtcweb/. [Accessed 06 11 2023].

[13] [Online]. Available: http://www.w3.org/2011/04/webrtc/ . [Accessed 06 11 2023].

[14] G. Suciu, S. Stefanescu, C. Beceanu and M. Ceaparu, "WebRTC role in real-time communication and video conferencing," in *2020 Global Internet of Things Summit (GIoTS)*, 2020.

[15] IETF, "RFC 8827 WebRTC Security Archiecture," 2021.

[16] IETF, "RFC 6455 The WebSocket Protocol," 2021.

[17] IETF, "RFC 3261 SIP: Session Initiation Protocol," 2002.

[18] ITU-T, "Infrastructure of audiovisual services – Systems and terminal equipment for audiovisual services – Packet-based multimedia communications systems: Recommendation ITU-T H.323," 2022.

[19] IETF, "RFC 8489 Session Traversal Utilities for NAT (STUN)," 2020.

[20] IETF, "RFC 5766 Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)," 2020.

[21] IETF, "RFC 8445 Interactive Connectivity Establishment (ICE)," 2019.

[22] IETF, "RFC 8866 SDP: Session Description Protocol," 2021.

[23] IETF, "RFC 3264 An Offer/Answer Model with the Session Description Protocol (SDP)," 2002.

[24] IETF, "RFC 8839 Session Description Protocol (SDP) Offer/Answer Procedures for Interactive Connectivity Establishment (ICE)," 2021.

[25] T. Stockhammer, "Dynamic Adaptive Streaming over HTTP – Standards and Design Principles," in *Proc. ACM Multimedia Systems (MMSys) Conference 2011*, San Jose, CA, USA, 2011.

[26] I. Sodagar, "The MPEG-DASH Standard for Multimedia Streaming over the Internet," *IEEE MultiMedia,* vol. 18, no. 4, 2011.

[27] R. Pantos and W. May, *HTTP Live Streaming. RFC 8216,* 2017.

[28] I. Viola, J. Jansen, S. Subramanyam, I. Reimat and P. Cesar, "VR2Gather: A Collaborative, Social Virtual Reality System for Adaptive, Multiparty Real-Time Communication," *IEEE MultiMedia,* vol. 30, no. 2, 2023.

[29] B. Taraghi, H. Hellwagner and C. Timmerer, "LLL-CAdViSE: Live Low-Latency Cloud-Based Adaptive Video Streaming Evaluation Framework," *IEEE Access,* vol. 11, 2023.

[30] SPIRIT Consortium, "Use Case Requirements, System Architecture and Interface Definition (First Version)," Project "Scalable Platform for Innovations on Real-time Immersive Telepresence", EC grant agreement 101070672, 2023.

[31] J. Santos, C. Wang, T. Wauters and F. D. Turck, "Diktyo: Network-Aware Scheduling in Container-based Clouds," *IEEE Transactions on Network and Service Management,* 2023.

[32] N. W. V. S. H. H. S. B. J. Y. a. R. T. S. Anmulwar, "Holosync: Frame synchronisation for multi-source holographic teleportation applications," *IEEE Transactions on Multimedia,* pp. 1-14, 2022.

[33] M. De Fré, J. van der Hooft, T. Wauters and F. De Turck, "Scalable MDC-Based Volumetric Video Delivery for Real-Time One-to-Many WebRTC Conferencing," in *ACM Multimedia Systems*, Bari, 2024.

[34] M. De Fré, J. van der Hooft, T. Wauters and F. De Turck, "Scalable MDC-Based WebRTC for Real-Time One-to-Many Volumetric Video Conferencing," *ACM Transactions on Multimedia Computing, Communications, and Applications,* 2024.

[35] J. van der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. Rondao Alface, T. Bostoen and F. De Turck, "HTTP/2-Based Adaptive Streaming of HEVC Video Over 4G/LTE Networks," *IEEE Communications Letters,* 2016.

[36] M. De Fré, J. van der Hooft, T. Wauters and F. De Turck, "Demonstrating Adaptive Many-to-Many Immersive Teleconferencing for Volumetric Video," in *ACM Multimedia Systems*, Bari, 2024.

[37] Wowza, "Video Streaming Latency Report," 2021. [Online]. Available: https://www.wowza.com/blog/2021-video-streaming-latency-report.

[38] K. E. Makkaoui, A. Beni-Hssane and A. Ezzati, "Can hybrid Homomorphic Encryption schemes be practical?," *5th International Conference on Multimedia Computing and Systems (ICMCS),,* vol. DOI: 10.1109/ICMCS.2016.7905580, 2016.

[39] "Intel: Software Guard Extensions (SGX)," [Online]. Available: https://www.intel.com/content/www/us/en/developer/tools/software-guard-extensions/overview.html. [Accessed 25 10 2022].

[40] "AMD: AMD Secure Encrypted Virtualization (SEV)," [Online]. Available: https://developer.amd.com/sev/. [Accessed 25 10 2022].

[41] "ARM: Trustzone," [Online]. Available: https://www.arm.com/technologies/trustzone-for-cortex-a. [Accessed 25 10 2022].

[42] "Trusted Computing Group," [Online]. Available: https://trustedcomputinggroup.org/resource/tpm-library-specification/. [Accessed 19 10 2022].

[43] "Intel: Intel® SGX Product Licensing," [Online]. Available: https://www.intel.com/content/www/us/en/developer/articles/technical/intel-sgx-product-licensing.html. [Accessed 01 11 2022].

[44] "Foreshadow: Breaking the Virtual Memory Abstraction with Transient Out-of-Order Execution," [Online]. Available: https://foreshadowattack.eu/. [Accessed 14 10 2022].

[45] "ÆPIC Leak: Architecturally Leaking Uninitialized Data from the Microarchitecture," [Online]. Available: https://aepicleak.com. [Accessed 14 10 2022].

[46] "Intel: Intel® Trust Domain Extensions. Whitepaper," [Online]. Available: https://cdrdv2.intel.com/v1/dl/getContent/690419. [Accessed 22 02 2023].

[47] Intel: Intel® Trust Domain Extensions - SEAM Loader (SEAMLDR) Interface Specification, Intel, March 2022.

[48] "Phoronix: Intel TDX Guest Attestation Support Merged For Linux 6.2," [Online]. Available: https://www.phoronix.com/news/Intel-TDX-Guest-In-Linux-6.2. [Accessed 06 03 2023].

[49] Intel, Intel: Intel® Trust Domain Extensions (Intel® TDX) Module Base Architecture Specification, Intel, September 2021.

[50] R. Buhren, Resource Control Attacks against Encrypted Virtual Machines, Berlin: PhD thesis, 2022.

[51] "Confidential Computing," [Online]. Available: https://sconedocs.github.io/workflows/. [Accessed 01 10 2022].

[52] "Gramine: Introduction to Gramine," [Online]. Available: https://gramine.readthedocs.io/en/latest/index.html. [Accessed 01 11 2022].

[53] "Microsoft Azure Attestation," [Online]. Available: https://learn.microsoft.com/en-us/azure/attestation/. [Accessed 03 11 2022].

[54] "Azure Confidential VM options on AMD," [Online]. Available: https://learn.microsoft.com/en-us/azure/confidential-computing/virtual-machine-solutions-amd. [Accessed 02 11 2022].

[55] "Azure/confidential-computing-cvm-guest-attestation," [Online]. Available: https://github.com/Azure/confidential-computing-cvm-guest-attestation. [Accessed 13 10 2022].

[56] "Google: Confidential Computing," [Online]. Available: https://cloud.google.com/confidential-computing. [Accessed 29 09 2022].

[57] "Amazon: AWS Nitro System," [Online]. Available: https://aws.amazon.com/ec2/nitro/?nc1=h_ls. [Accessed 13 10 2022].

[58] "Intel: Intel® NUC 9 Pro Kit - NUC9VXQNX," [Online]. Available: https://www.intel.com/content/www/us/en/products/sku/190108/intel-nuc-9-pro-kit-nuc9vxqnx/specifications.html. [Accessed 01 11 2022].

[59] "SGX101 Sample Code, Github repository," [Online]. Available: https://github.com/sangfansh/SGX101_sample_code. [Accessed 01 11 2022].

[60] "Launch security with AMD SEV," [Online]. Available: https://libvirt.org/kbase/launch_security_sev.html. [Accessed 26 10 2022].

[61] "Gramine: Quick Start," [Online]. Available: https://gramine.readthedocs.io/en/latest/quickstart.html. [Accessed 01 11 2022].

[62] "Gramine Github repository," [Online]. Available: https://github.com/gramineproject/gramine.git. [Accessed 01 11 2022].

[63] Google, [Online]. Available: https://cloud.google.com/compute/docs/instances/verifying-instance-identity#verifying. [Accessed 13 12 2022].

[64] "Eventscase: Data Security for Virtual & Hybrid Events," [Online]. Available: https://eventscase.com/blog/data-security-for-virtual-and-hybrid-events. [Accessed 07 02 2023].

[65] "Google: Verifying the identity of instances," [Online]. Available: https://cloud.google.com/compute/docs/instances/verifying-instance-identity#verifying. [Accessed 13 10 2022].

[66] Intel. [Online]. Available: https://www.intel.com/content/dam/www/public/us/en/documents/datasheets/trusted-execution-technology-server-platforms-matrix.pdf. [Accessed 19 10 2022].

[67] SPIRIT Consortium, "SPIRIT Open Call Toolkit," Project "Scalable Platform for Innovations on Real-time Immersive Telepresence", EC grant agreement 101070672, 2023.

[68] T. Frosch, C. Mainka, C. Bader, F. Bergsma, J. Schwenk and T. Holz, "How Secure is TextSecure?," *IEEE European Symposium on Security and Privacy (EuroS&P),* vol. doi: 10.1109/EuroSP.2016.41., pp. 457-472, 2016.

[69] SPIRIT Consortium, "Innovation Platform Enablers (Second Version)," Project "Scalable Platform for Innovations on Real-time Immersive Telepresence", EC grant agreement 101070672, 2024.

6 APPENDIX

A.1 PRACTICAL EXPERIMENTATION WITH TRUSTED EXECUTION TECHNOLOGIES FOR CLOUD AND EDGE WORKLOADS

A.1.1 Practical Experiences with Intel SGX

We have used an Intel NUC 9 Pro Kit (NUC9VXQNX) with Intel Xeon E-2286M Processor for local testing [59]. As described in [TESTDCAP], the output of "`cpuid | grep -i sgx`" shows that Intel SGX (in the variant of SGX1) and DCAP (because launch config is "true") is supported:

```
    SGX: Software Guard Extensions supported = true
    SGX_LC: SGX launch config supported      = true
 Software Guard Extensions (SGX) capability (0x12/0):
    SGX1 supported                           = true
```

For testing we have used a simple HelloWorld application that shows the split of an SGX-enabled application into a "normal" Linux executable while the SGX-protected part is implemented as a shared object [60].

The main program (excerpt) looks like this:

```
[...]
/* OCall functions */
void ocall_print_string(const char *str)
{
    /* Proxy/Bridge will check the length and null-terminate
     * the input string to prevent buffer overflow.
     */
    printf("%s", str);
}


/* Application entry */
int SGX_CDECL main (int argc, char *argv[])
{
    (void)(argc);
    (void)(argv);


    /* Initialize the enclave */
    if(initialize_enclave() < 0) {
        printf("Enter a character before exit ...\n");
        getchar();
        return -1;
    }

    printf_helloworld(global_eid);

    /* Destroy the enclave */
    sgx_destroy_enclave(global_eid);

    return 0;
}
```

The main program initializes the enclave (code not shown) and then calls the "`printf_helloworld`" function from the enclave shared object. The source code for the enclave is this:

```
#include "Enclave_u.h"
#include <errno.h>

typedef struct ms_ocall_print_string_t {
        const char* ms_str;
} ms_ocall_print_string_t;

static sgx_status_t SGX_CDECL Enclave_ocall_print_string(void* pms)
{
        ms_ocall_print_string_t* ms = SGX_CAST (ms_ocall_print_string_t*, pms);
        ocall_print_string(ms->ms_str);

        return SGX_SUCCESS;
}

static const struct {
        size_t nr_ocall;
        void * table [1];
} ocall_table_Enclave = {
        1,
        {
                (void*)Enclave_ocall_print_string,
        }
};
sgx_status_t printf_helloworld(sgx_enclave_id_t eid)
{
        sgx_status_t status;
        status = sgx_ecall(eid, 0, &ocall_table_Enclave, NULL);
        return status;
}
```

The "`printf_helloworld`" function still runs in the unprotected context of the main application. It uses an SGX API function "`sgx_ecall`" to call into the protected SGX enclave, where the "`Enclave_ocall_print_string`" function is executed. Since SGX enclaves cannot do I/O (input/output) directly, the enclave must call back to the main program to print out the actual HelloWorld message. Finally, the "`ocall_print_string`" function of the main program does the output using a standard C function (which would not be available to the enclave).

The output of this remarkably simple program is:

```
$. /app
Hello World
```

This example shows the overhead (while glossing over many details) of implementing secure functionality using SGX technology. Additionally, the example currently runs in debug mode with no actual protections. A commercial license would be required to implement encryption and secure remote attestation.

A.1.2 Practical Experiences with AMD-SEV

For our experiments we used a Dell PowerEdge R7525 AMD EPYC 7313 16 core dual processor system that supports AMD SEV-SNP. In order to enable full SEV-SNP functionality we had to enable SNP and several options in the BIOS which required several reboots (see Figure 51).

Secure Memory Encryption	Enabled ⬍
Minimum SEV non-ES ASID	16
Secure Nested Paging	Enabled ⬍
SNP Memory Coverage	Enabled ⬍
Transparent Secure Memory Encryption	Enabled ⬍

FIGURE 51: BIOS SETTINGS FOR AMD EPYC SERVER TO FULLY ENABLE SEV-SNP

After these BIOS changes, a "`dmesg |grep SEV`" yields the following output:

```
[    1.633314] ccp 0000:22:00.1: SEV API:1.52 build:4
[    4.841318] SEV supported: 494 ASIDs
[    4.841318] SEV-ES supported: 15 ASIDs
```

This means that SEV is enabled in the hardware and supported by the Linux kernel. Specifically, 494 confidential VMs and 15 VMs with encrypted state (SEV-ES) are supported. SEV-ES means that all CPU register contents are encrypted when a VM stops running.

We use KVM/libvirt as the virtualization system to run confidential VMs. The "`virsh domcapabilities`" command also shows the enabled SEV Support:

```
<sev supported='yes'>
  <cbitpos>51</cbitpos>
  <reducedPhysBits>1</reducedPhysBits>
  <maxGuests>494</maxGuests>
  <maxESGuests>15</maxESGuests>
<cpu0Id>FYI7dYcBx6MYI77qoLypLwVzWn50s33V/IlopFl7VnjS5Fpn4QtXn15L6PfCCWo5ekEaMEAyn/T3
WaoqpbPtiA==</cpu0Id>
  </sev>
```

The following command installs the confidential VM with SEV enabled directly from an online Centos 9 repository (for more details refer to [61]):

```
virt-install --name centos-sev-es \
    --location                https://ftp.uni-bayreuth.de/linux/centos-stream/9-
stream/BaseOS/x86_64/os \
    --disk size=20 --network=bridge=virbr0, model=virtio,rom.bar=off \
    --vcpus 4 --memory 4096 --noautoconsole --events on_reboot=destroy \
    --machine q35 --memtune hard_limit=4563402 --launchSecurity sev,policy=0x07
\
    --boot firmware=efi  --vnc --serial pty
```

The install can continue using the `virt-manager` console view (refer to Figure 52).
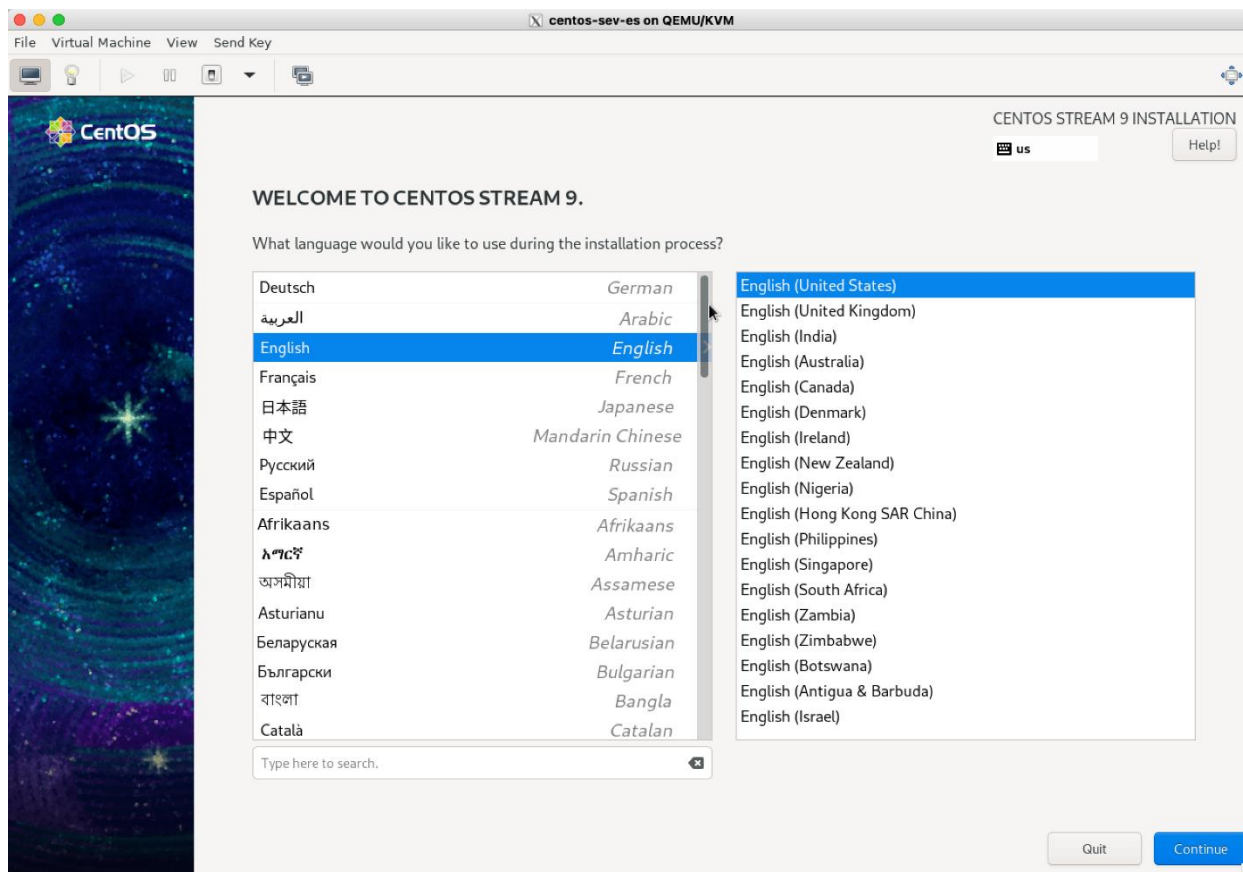
FIGURE 52: CENTOS 9 INSTALLATION STARTS IN AMD SEV-SNP VM

In the guest VM one check whether SEV is active:

```
$ dmesg | grep SEV
[    0.075000] AMD Memory Encryption Features active: SEV SEV-ES
```

Following these steps, it has been possible to install a production ready encrypted VM without any interaction with AMD.

A.1.3 Practical Experimentation with Software Frameworks

A.1.3.1 Practical Example with Gramine

After installing the Gramine binary distribution for Centos 8 (which also works with some tweaks on our Fedora 36 development platform), we can follow the Gramine quick start instructions [62].

First, a private key must be initially created for enclave signing:

```
$ gramine-sgx-gen-private-key
wrote RSA 3072 private key to /home/hofmannp/.config/gramine/enclave-key.pem
```

We then install the Gramine GitHub repository [63] to access the HelloWorld application:

```
git clone --depth 1   https://github.com/gramineproject/gramine.git
```

The provided HelloWorld application is quite simple – it would also be possible to directly compile it and run it without the Gramine environment:

```c
#include <stdio.h>

int main(void) {
    printf("Hello, world\n");
    return 0;
}
```

The HelloWorld application can be built with "`make SGX=1`" which yields the following output:

```
gramine-sgx-sign \
     --manifest helloworld.manifest \
     --output helloworld.manifest.sgx
Attributes:
    size:        0x10000000
    thread_num:  4
    isv_prod_id: 0
    isv_svn:     0
    attr.flags:  0x6
    attr.xfrm:   0x3
    misc_select: 0x0
SGX remote attestation:
    None
Memory:
    000000000fffd000-0000000010000000 [REG: R--] (manifest) measured
    000000000ffdd000-000000000fffd000 [REG:RW-] (ssa) measured
    000000000ffd9000-000000000ffdd000 [TCS: ---] (tcs) measured
    000000000ffd5000-000000000ffd9000 [REG:RW-] (tls) measured
    000000000ff95000-000000000ffd5000 [REG:RW-] (stack) measured
    000000000ff55000-000000000ff95000 [REG:RW-] (stack) measured
    000000000ff15000-000000000ff55000 [REG:RW-] (stack) measured
    000000000fed5000-000000000ff15000 [REG:RW-] (stack) measured
    000000000fec5000-000000000fed5000 [REG:RW-] (sig_stack) measured
    000000000feb5000-000000000fec5000 [REG:RW-] (sig_stack) measured
    000000000fea5000-000000000feb5000 [REG:RW-] (sig_stack) measured
    000000000fe95000-000000000fea5000 [REG:RW-] (sig_stack) measured
    000000000f87a000-000000000f8bd000 [REG:R-X] (code) measured
    000000000fabd000-000000000fe95000 [REG:RW-] (data) measured
    0000000000010000-000000000f87a000 [REG:RWX] (free)
Measurement:
    a970c871f7117e932f5ca62dd9ecce86de06742329657443dd6d561f66c07ac9
gramine-sgx-get-token \
     --output helloworld.token --sig helloworld.sig
Attributes:
    mr_enclave:  a970c871f7117e932f5ca62dd9ecce86de06742329657443dd6d561f66c07ac9
    mr_signer:   a6fc0183e328c7ac2a296caed3075ce296c2424ee7e03cc8943aef870122543f
    isv_prod_id: 0
    isv_svn:     0
    attr.flags:  0000000000000006
    attr.xfrm:   0000000000000007
    mask.flags:  ffffffffffffffff
    mask.xfrm:   fffffffffff9ff1b
    misc_select: 00000000
    misc_mask:   ffffffff
    modulus:     5b7ecb539f3dd11431f133bbaaf3199a...
    exponent:    3
    signature:   e45e74864aa905f5a820c8a7c0198d20...
    date:        2022-11-01
```

Running this sample application using "`gramine-sgx helloworld`" gives the expected output:

```
Gramine is starting. Parsing TOML manifest file, this may take some time...
------------------------------------------------------------------------------
---------------------------------
Gramine detected the following insecure configurations:

  - sgx.debug = true                         (this is a debug enclave)

Gramine will continue application execution, but this configuration must not be used
in production!
------------------------------------------------------------------------------
---------------------------------

Hello, world
```
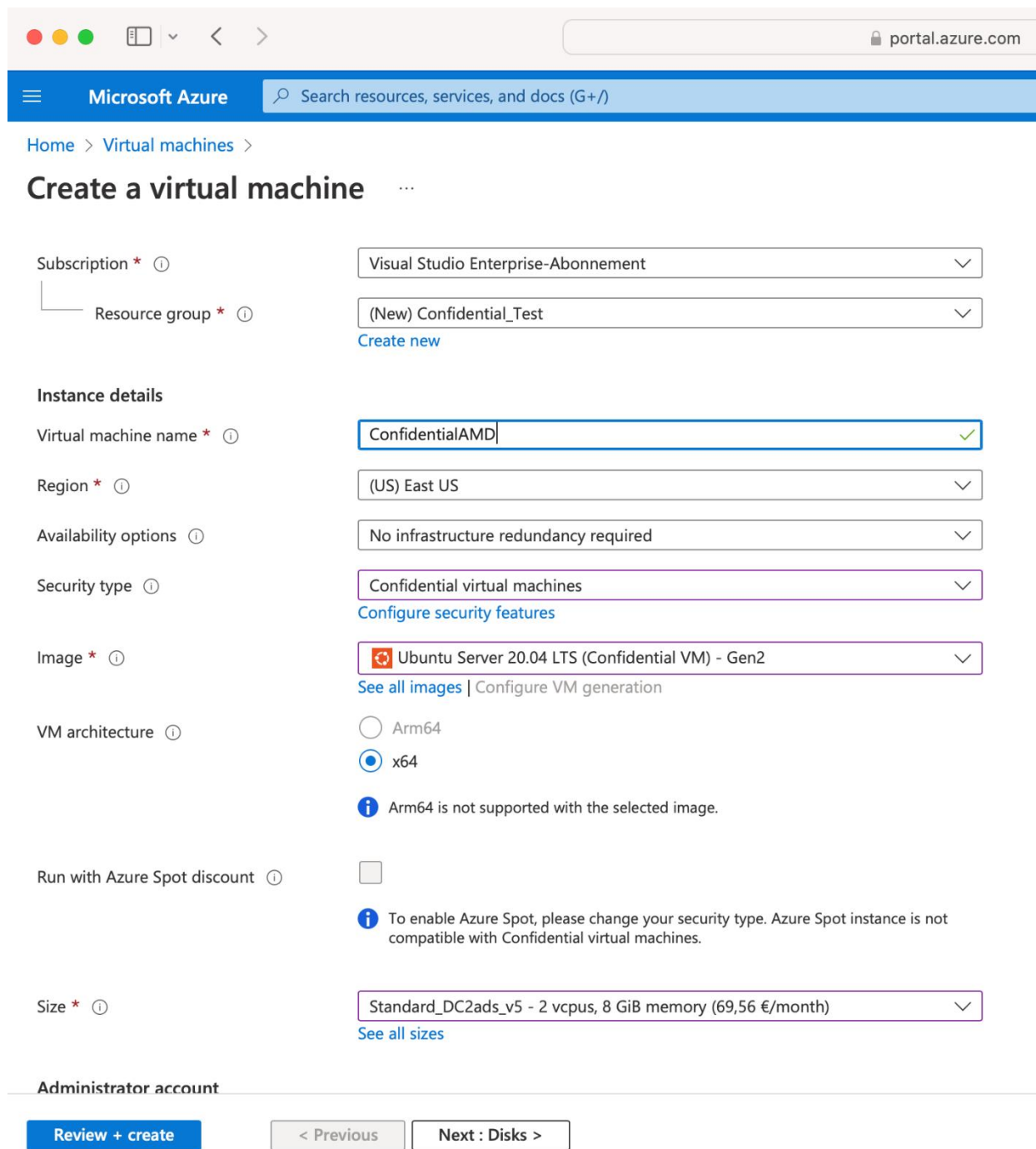
As with the SGX 101 (see section A.1.1) example, this demo runs in SGX debug mode. Compared to the SGX example code, however, no changes to the sample application and no complex glue code were necessary to wrap it into an SGX enclave.

A.1.4 Experimentation with Trusted Execution Environments by Public Cloud Providers

A 1.4.1 Microsoft Azure

A 1.4.1.1 AMD SEV-SNP

Figure 53 shows the creation screen for an AMD-based confidential VM in Microsoft Azure.

FIGURE 53: CREATION OF AMD-BASED CONFIDENTIAL VM IN AZURE.

This VM is based on the "DC2ads_v5" machine type, which is an EPYC 7763v processor that supports SEV-SNP. For this instance, we have chosen the supported Linux version (Ubuntu Server 20.04). After a few minutes of deployment, the confidential Linux system is ready to use.
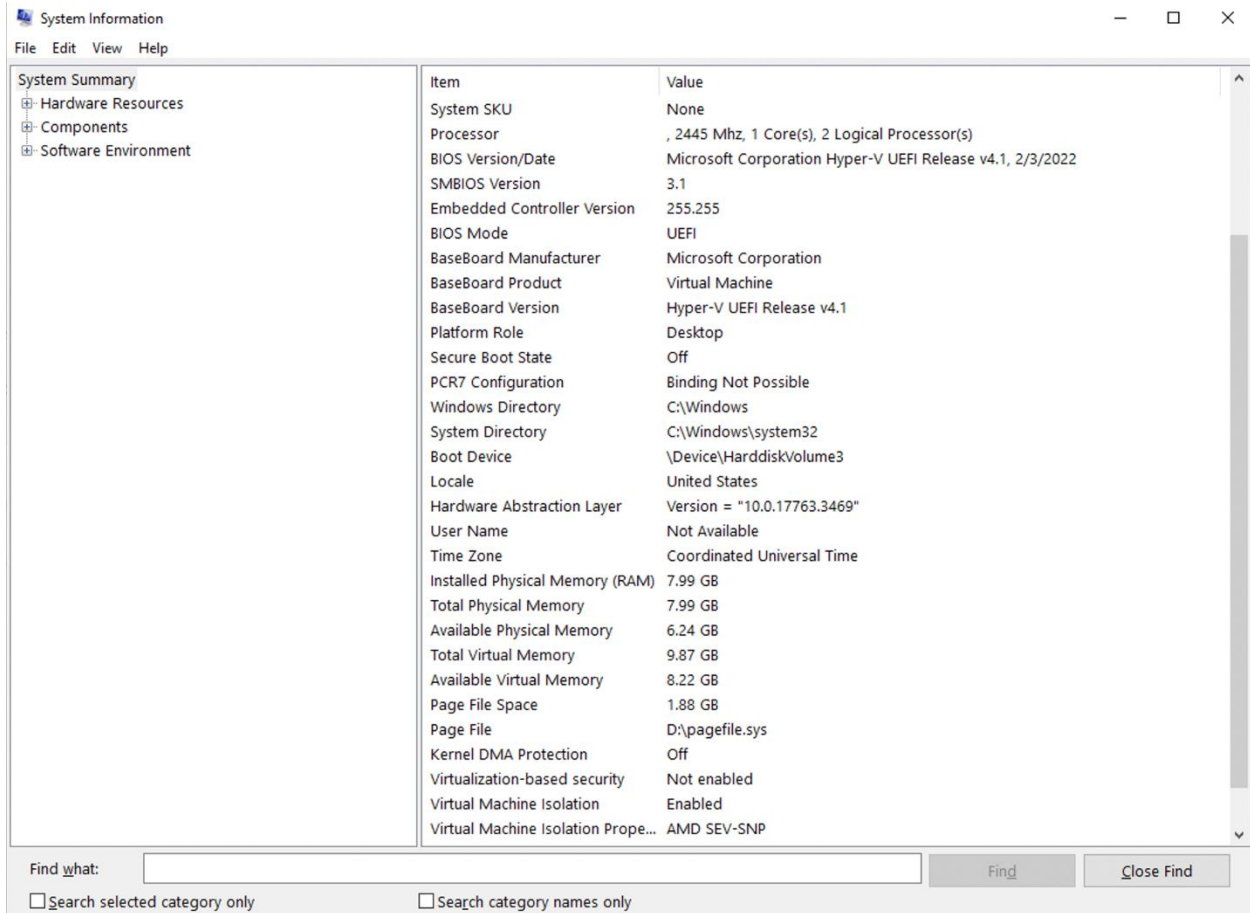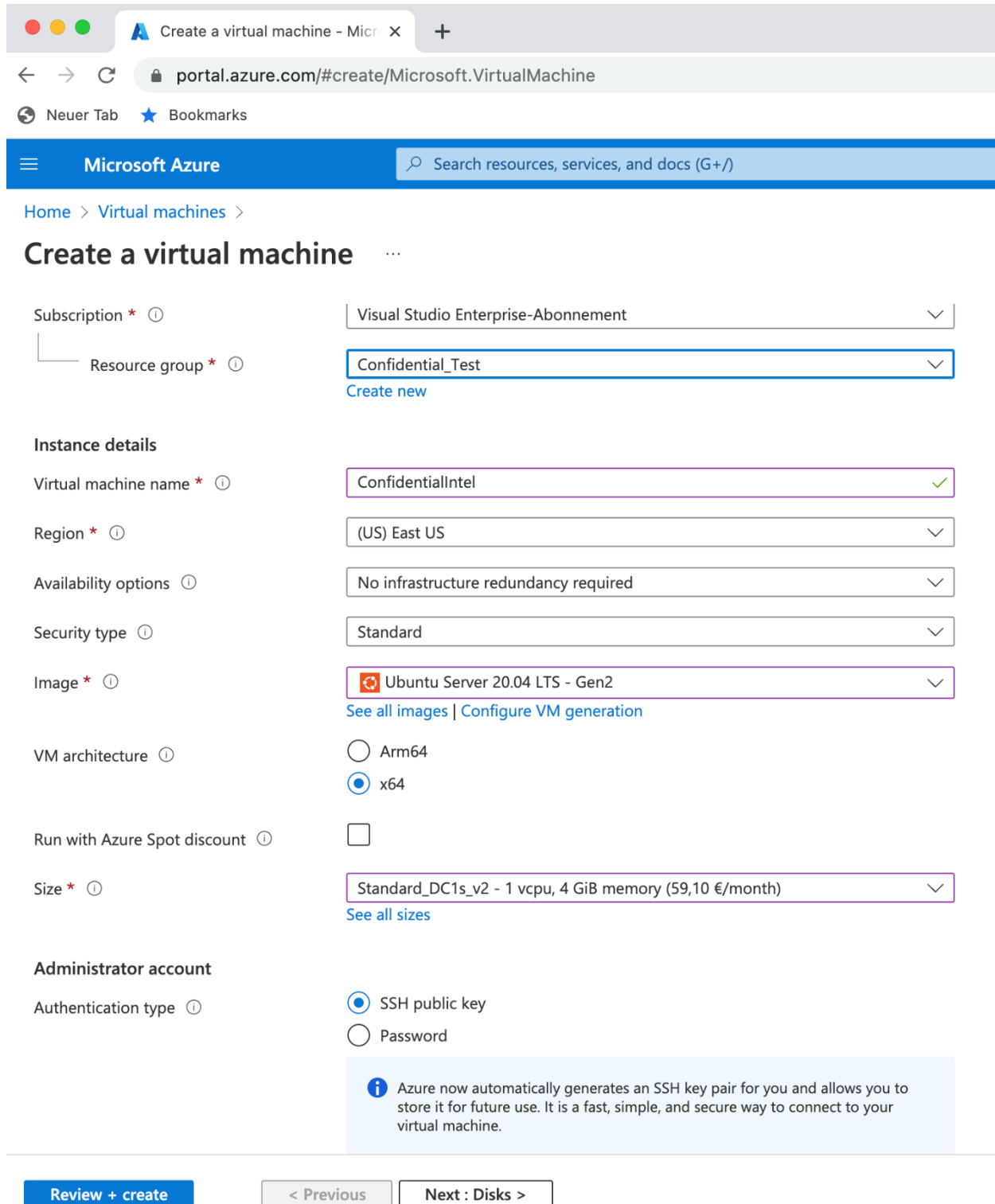
FIGURE 54: MSINFO OUTPUT SHOWING SEV-SNP GUEST SUPPORT IN WINDOWS.

Alternatively, Azure allows the deployment of Windows Server VM guests. Figure 54 shows the output of the `msinfo` program running in the Windows Server VM, showing that SEV-SNP is enabled in the guest (see two lines at the bottom).

A 1.4.1.2 Intel SGX

Figure 55 shows the creation screen in the Azure portal for the creation of an Intel SGX-based confidential VM that supports SGX enclaves.

FIGURE 55: CREATION OF INTEL-BASED CONFIDENTIAL VM IN AZURE.

The output of "`cpuid | grep -i sgx`" shows that SGX is now active on this newly created VM:

```
SGX: Software Guard Extensions supported = true
SGX_LC: SGX launch config supported     = true
Software Guard Extensions (SGX) capability (0x12/0):
SGX1 supported                          = true
```
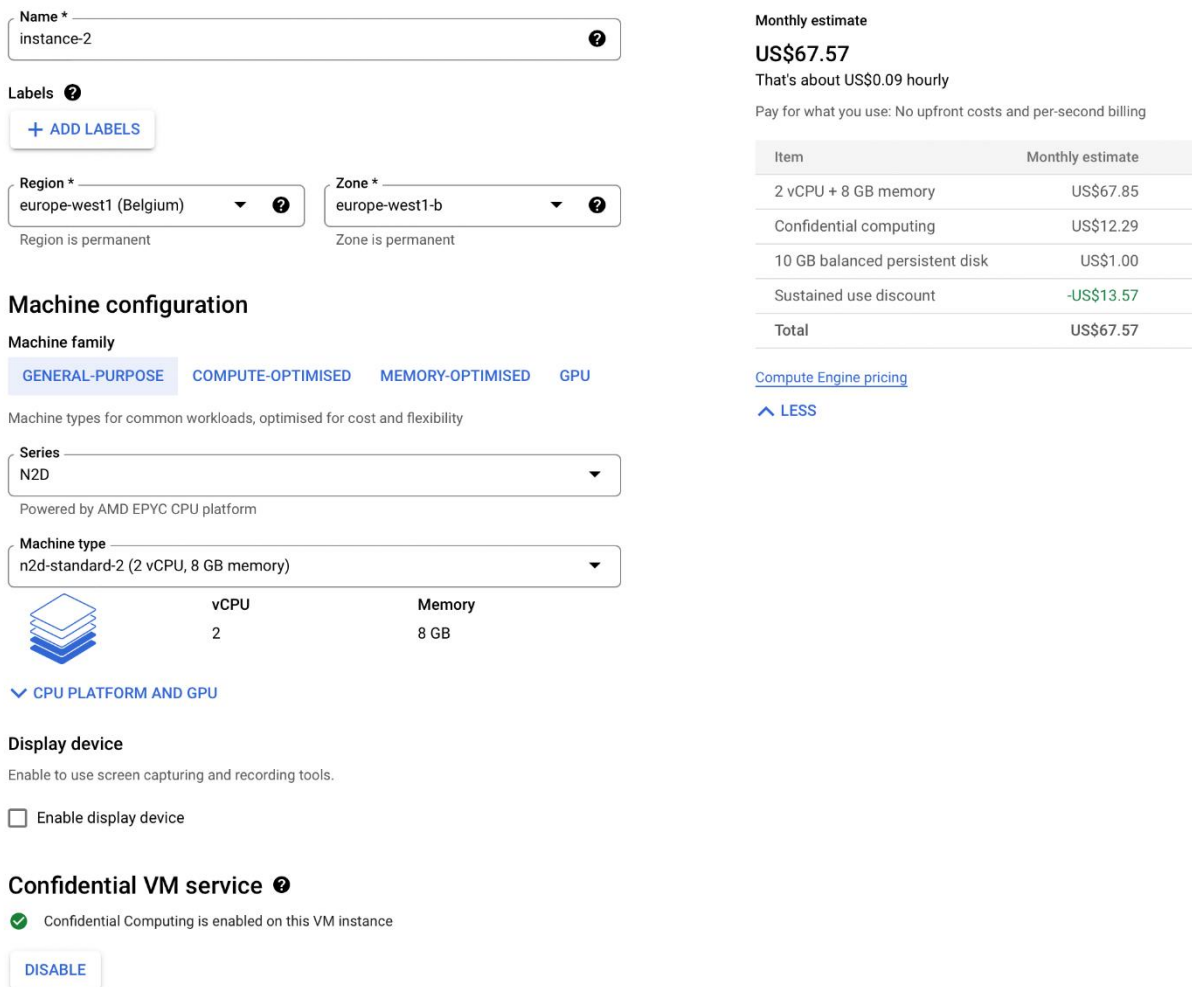
```
    SGX2 supported                        = false
    SGX ENCLV E*VIRTCHILD, ESETCONTEXT    = false
    SGX ENCLS ETRACKC, ERDINFO, ELDBC, ELDUC = false
  SGX attributes (0x12/1):
```

One can deploy SGX-enabled applications on this platform.

A.1.4.2 Google Cloud

In order to create a confidential VM in Google Cloud, one must press the button "Create a VM" on the Google cloud dashboard.

FIGURE 56: GOOGLE CLOUD CONFIDENTIAL VM CREATION.

As you can see in Figure 56, it is just necessary to press "Enable" to make the VM confidential. This setting takes a Google-supplied Linux VM template and instantiates it with default measurements and attestations.

The Google log console allows to view regular events to check the integrity of the started confidential VM. A sample integrity event looks like this:

```
{
  "insertId": "0",
  "jsonPayload": {
    "sevLaunchAttestationReportEvent": {
```

Funded by the Horizon Europe
Framework Programme of the European Union

```
    "sevPolicy": {
      "sevOnly": true,
      "minApiMajor": 0,
      "domainOnly": false,
      "minApiMinor": 0,
      "esRequired": false,
      "sendAllowed": true,
      "keySharingAllowed": false,
      "debugEnabled": false
    },
    "integrityEvaluationPassed": true
  },
  "@type": "type.googleapis.com/cloud_integrity.IntegrityEvent",
  "bootCounter": "0"
},
"resource": {
  "type": "gce_instance",
  "labels": {
    "zone": "europe-west1-b",
    "instance_id": "8783936653038850848",
    "project_id": "pure-environs-365309"
  }
},
"timestamp": "2022-10-13T06:19:10.207679246Z",
"severity": "NOTICE",
"logName": "projects/pure-environs-
365309/logs/compute.googleapis.com%2Fshielded_vm_integrity",
"receiveTimestamp": "2022-10-13T06:19:12.221460008Z"
}
```

On the command line of the VM, the state of AMD SEV can be checked on the command line:

```
dmesg | grep SEV | head
```

This yields the following output on the machine created as described above:

```
[    0.154743] AMD Secure Encrypted Virtualization (SEV) active
```

Of course, this is no cryptographic proof that the remote VM is genuine. Google implements a system with signed JWT tokens to verify the integrity of remote computing resources, described here *[64]*.