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

CHARITY
Cloud for Holography and Augmented RealITY

Cloud for Holography and Cross Reality

D1.3: CHARITY architecture design and specification

Version: v1.0

Deliverable type	R (Document, report)
Dissemination level	PU (Public)
Due date	31/12/2021
Submission date	27/12/2021
Lead editor	Tarik Taleb (ICT-FI)
Authors	Abderrahmane Boudi (ICT-FI), Antonios Makris (HUA), Theodoros Theodoropoulos (HUA), Konstantinos Tserpes (HUA), Laura Sande Alonso (PLEX), Thomas Loven (PLEX), Peter Gray (CS), Martin Vasev (CS), Luis Rosa (ONE), Marco Di Girolamo (HPE), Alessandro Romussi (HPE), Elena Spatafora (HPE), Thu Le Pham (UTRC), Michael McElligott (UTRC), Enrico Zschau (SRT), Aravindh Raman (TID), Ferran Diego Andilla (TID), Antonis Protopsaltis (ORAMA), Manos Kamarianakis (ORAMA),
Reviewers	Phil Harris (UTRC)
Work package, Task	WP1, T1.2
Keywords	Architecture Design, Open Source, Use Case Validation, Immersive Services

Abstract

This deliverable introduces the CHARITY architecture to support future XR applications. First, it surveys the different technologies, standards, and system architectures that inspired the design work of the CHARITY architecture. Where possible, the relation to the CHARITY architecture is highlighted.

The CHARITY architecture is then presented, describing its planes and its components. The feasibility of the CHARITY architecture is then discussed, by mapping its envisioned functionalities to potential open sources and tools. The architecture is then validated against the project use cases. The potential stakeholders are also identified.

Document revision history

Version	Date	Description of change	List of contributor(s)
v0.1	26/04/21	Add ToC	All
v0.2	07/12/21	Final version of section 1-3	All
v0.3	10/12/21	Add section 4 final version	ICT-FI + UC owners
v0.4	12/12/21	Revision	Abderrahmane Boudi
v0.5	16/12/21	Revision	Tarik Taleb
v0.9	17/12/21	Final editing and issue for GA approval	Uwe Herzog, Anja Köhler
v1.0	22/12/21	Minor corrections, list of abbreviations added, Executive Summary extended	Tarik Taleb, Abderrahmane Boudi, Anja Köhler, Uwe Herzog

Disclaimer

Acknowledgment

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

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

Executive Summary

This deliverable introduces the CHARITY architecture to support future XR applications. After introducing the use cases of the project along with their requirements in deliverable D1.2, this deliverable introduces an architecture that can support the use cases and meet their requirements.

The methodology adopted in the research work conducted here to devise the architecture is as follows. First, we conducted an extensive survey about the different technologies, standards, and system architectures relevant to the project. That ultimately inspired the design work of the CHARITY architecture. This deliverable presents the different relevant technologies, standards, and system architectures, and where possible, the relation to the CHARITY architecture is highlighted. The intention beneath this step is to ensure that the proposed architecture would benefit from the latest versions of the standards and also to make sure it is compatible with the XR technologies. In short, the CHARITY architecture draws its inspiration from ETSI, 3GPP standards and also from other European projects' architectural work.

The CHARITY architecture is then presented, describing its planes and its components. These are architected in a way to ensure an edge-cloud continuum of resources that the XR services can readily consume. Another focal point of the architecture is to provide a high level of automation where the XR services would react and adapt to various changes that arise from the underlying infrastructure or from the XR services themselves. This is done by providing an automation loop that is heavily inspired from ETSI ZSM framework. This architecture also supports and makes use of the DevOps concept that is tailored specifically to support the deployment and the life cycle management of XR services on top of the CHARITY infrastructure.

The feasibility of the CHARITY architecture is subsequently discussed, by mapping between the most relevant surveyed open sources and tools with the key components of the CHARITY architecture. In the initial stage, the focus was on understanding existing open sources components and tools to fulfill the vision of CHARITY as a distributed platform across multiple domains. Namely, we targeted tools to expose and instantiate resources spanning distinct domains (i.e., cross-domain resource management and orchestration tools), how to monitor them (i.e., service-meshes and their related monitoring components) and finally, how to interconnect CHARITY components through efficient integration fabrics, both within and across domains.

Finally, the architecture is validated against the project use cases and the potential stakeholders are also identified. In fact, the validation consists into providing a walkthrough, for each use case, on how the XR service can be integrated into the CHARITY platform. It showcases all the phases that the XR service will pass through, starting from its development, to its deployment and exploitation, until its decommissioning. This proved the validity of the architecture and demonstrated that the envisioned architecture can support XR services. This exercise also offered insights on the integration work envisioned in WP4.

This deliverable is one of the foundations that would support the CHARITY project during its entire lifetime. As it can be seen in the conclusion, it sets up a framework for the other technical WPs (i.e., WP2, WP3, and WP4) which gives some guidelines on the architectural components to be developed and also the integration work needed for the use case owners in order to fully take advantage of the proposed architecture.

Table of Contents

List of Figures

List of Tables

Abbreviations

3D	Three-Dimensional
3DOF	Three Degrees of Freedom
3GPP	Third Generation Partnership Project
5G NF	Fifth Generation Network Function
5GMSd	5G Media Streaming downlink
6DOF	Six Degrees of Freedom
ACT	Actuation functions
ADT	Application Description Template
AE	Analytics Engine
AMF	Application Management Framework
API	Application Programming Interface
AR/VR	Augmented Reality/Virtual Reality
ARF	Augmented Reality Framework
AS	Assisted System
ASP	Application Service Provider
BRNN	Bidirectional Recurrent Neural Network
BSS	Business Support Systems
CFS	Customer Facing Service
CI/CD	Continuous Integration/Continuous Delivery
CIR	Container Image Registry
CISM	Container Infrastructure Service Management
CNCF	Cloud Native Computing Foundation
COLA	Cloud Orchestration at the Level of Application
CP	Control Plane
CPU	Central Processing Unit
CU	Centralized Unit
DAI	Distributed AI
DASH	Dynamic Adaptive Streaming over HTTP
DE	Decision Engine
DHT	Distributed Hash Table
DNS	Domain Name System
DU	Distributed Unit
E2E	End-to-End
EAS	Edge Application Server
ECSP	Edge Computing Service Provider
EM	Element Management
EMS	Element Management System
ENI	Experiential Network Intelligence

ETSI	European Telecommunications Standards Institute
FCAPS	Fault, Configuration, Accounting, Performance, and Security management
glTF™	GL Transmission Format
gNB	5G radio base station
GPU	Graphics Processing Unit
HA	Holographic Assistant
HEVC	High Efficiency Video Coding
HMD	Head-Mounted Display
IFTTT	If This Then That
ISG	Industry Specification Group
ISM	In-Slice Management
ISOBMFF	ISO Base Media File Format
KPI	Key Performance Indicator
LCM	Lifecycle Management
LEAP	Latency-controlled End-to-End Aggregation Protocol
LFU	Least Frequently Used
MaaS	Management as a Service
MANO	NFV Management and Orchestration
MAPE-K	Monitor, Analyse, Plan, Execute
MAR	Mixed and Augmented Reality
MARS	Mixed and Augmented Reality System
MEC	Multi-access Edge Computing
MEO	Multi-access Edge Orchestrator
mLFU	Multiple-factors LFU
MMT	MPEG Media Transport
MNO	Mobile Network Operator
MPE	Media Processing Entity
MPEG	Moving Picture Experts Group
MR	Mixed Reality
MS	Monitoring System
NBMP	Network-based Media Processing
NEF	5G Network Exposure Function
NFV	Network Function Virtualisation
NFVI	Network Function Virtualisation Infrastructure
NFVO	Network Functions Virtualisation Orchestrator
NS	Network Service
NSO	Network Service Orchestrator
OMAF	Omnidirectional MediA Format
OODA	Observe, Orient, Decide, Act
O-RAN	Open Radio Access Network

ORPC	Open RAN Policy Coalition
OSM	Open-Source MANO
OSS	Operations Support System
P2P	Peer-To-Peer
PCDS	Point Cloud Data Set
PCF	5G Policy Control Function
PNF	Physical Network Function
PoP	Point of Presence
PTAM	Parallel Tracking and Mapping
QoE	Quality of Experience
QoS	Quality of Service
RAN	Radio-Access Network
RGB	Red Green Blue
RIC	RAN Intelligent Controller
RO	Resource Orchestrator
RT	Real Time
RU	Radio Unit
SDN	Software Defined Networking
SD-SEC	AI-driven Software Defined Security
SECaaS	Security as a Service
SFC	Service Function Chain
SLA	Service-level agreement
SLAM	Simultaneous localization and mapping
SMD	Security Management Domains
SSLA	Security Service Level Agreement
TIP	Telecom Infra Project
UE	User Equipment
UP	User Plane
UPF	User Plane Function
VIM	Virtual Infrastructure Manager
VM	Virtual Machine
VNF	Virtual Network Function
VNFM	VNF Manager
V-PCC	Video-based Point Cloud Compression
VR	Virtual Reality
vRAN	Virtualized Radio Access Network
WIM	WAN Infrastructure Manager
YCSB	Yahoo Cloud System Benchmark
ZSM	Zero-touch network & Service Management

1 Introduction

1.1 Objectives

This deliverable sums up the outcome of the research efforts carried out by all the CHARITY project partners within Task 1.2 of the project, and that is towards designing the CHARITY architecture. The architecture will be used as a reference for all tasks of WP2 and WP3, as well as for the integration activities of WP4.

1.2 Scope

This deliverable introduces the CHARITY architecture that puts together the key enablers to support future XR applications and to cope with the relevant challenges, considering the short-comings of existing technologies and the ongoing innovations in various fields.

In particular, this deliverable contributes to the state-of-art in the following ways:

- It defines a platform architecture that leverages existing standards to support XR.
- It demonstrates the feasibility of the proposed architecture by mapping its functionalities to existing tools and open sources.
- It validates the architecture against the project use cases, categorized in a) Real-time Holographic Applications, b) Immersive Virtual Training and c) Mixed Reality Interactive Applications.

1.3 Structure of the document

The deliverable is structured in the following fashion. Section 2 provides a detailed survey on most technologies, standards, open sources, and EU project architectures that relate to/inspired the design of the CHARITY architecture. The CHARITY architecture is introduced in Section 3. To demonstrate the feasibility of the CHARITY architecture, and to validate the concept, Section 3 also introduces potential tools and open sources, and maps them to the main functionalities of the architecture. The CHARITY architecture is validated against the envisioned use cases of the project in Section 4. Section 4 also discusses the potential stakeholders. Finally, the deliverable concludes in Section 5.

2 Related Work

This section serves the purpose of introducing — in a survey style — the different technologies, standards, and system architectures that inspired the design work of the CHARITY architecture. Where possible, the relation to the CHARITY architecture is highlighted. The CHARITY architecture is presented in Section 3.

2.1 ETSI Multi Access Edge Computing

Recently, the Edge computing paradigm has been considered as a key enabler for addressing the increasing strict requirements of next-generation applications. Contrary to (traditionally centralized) Cloud Computing, in Edge Computing the computational resources are placed physically closer to the end users into the so-called network edge. Amongst many others, this has the benefit of reducing the latency times. Moreover, the amount of data in-transit towards remote clouds is significantly reduced and the data processing occurs near to the data sources, ultimately expanding the possibilities for more delay-sensitive and high-bandwidth applications that would not be feasible using cloud and far remote processing alone.

The Multi-access Edge Computing (MEC) is a European Telecommunications Standards Institute (ETSI) — Industry Specification Group (ISG) standardization initiative within the telecommunication industry to specify how the Radio-Access Network (RAN) of telecommunication operators can be leveraged to realize the principles of Edge computing. MEC intends to provide a consistent path where multiple third parties, including service providers, can make use of the last mile of telecommunication operator networks and infrastructure to deploy their services and applications, in the same way they would do it in an IT environment. This opens a variety of new business opportunities for all parties. Telecommunication providers have more ways to capitalize their infrastructure, traditional cloud providers can expand their offer to a new range of services leveraging the computation resources of telecommunication operators, whereas next-generation application developers can create innovative applications specifically tailored for such ultra-low latency environments (i.e., the MEC applications).

The ETSI MEC ISG organized the MEC framework into three levels: system, host, and network levels. Figure 1 provides the generic reference architecture of a MEC system, its functional elements, and the reference points between them.

The host level includes the MEC host itself, which is comprised of the MEC applications, the Virtualisation Infrastructure and the MEC platform, and their management counterparts (i.e., the Virtualisation Infrastructure Manager (VIM) and the MEC platform Manager). On the other hand, the MEC platform is defined by all the functional blocks that allow to consume and provide services (e.g., service registry, DNS handling and traffic control in-general).

The system level is composed of a Multi-access edge orchestrator (MEO) which has the responsibility of the overall view of the MEC system, hosts and applications and handling the life-cycle of each MEC application, including for instance the selection of the appropriate MEC hosts for a given application taking into considerations specific constrains such as latency or resource availability. The system level is also composed of the Operations Support System (OSS) of telecommunication operator and the respective Customer Facing Service (CFS), which together are responsible for instantiating and forwarding application requests to the MEO.

Finally, the network level refers to the external and network related entities (e.g., 3GPP Network, Local Network, and External Network).

The MEC reference architecture was also designed taking into consideration the generic Network Function Virtualisation (NFV) reference architecture. Likewise, the CHARITY framework, which was also inspired on previous works and specifications, is also compatible with the NFV reference architecture. Thus, CHARITY, as a flexible approach, can also support as well as benefit from the MEC paradigm.

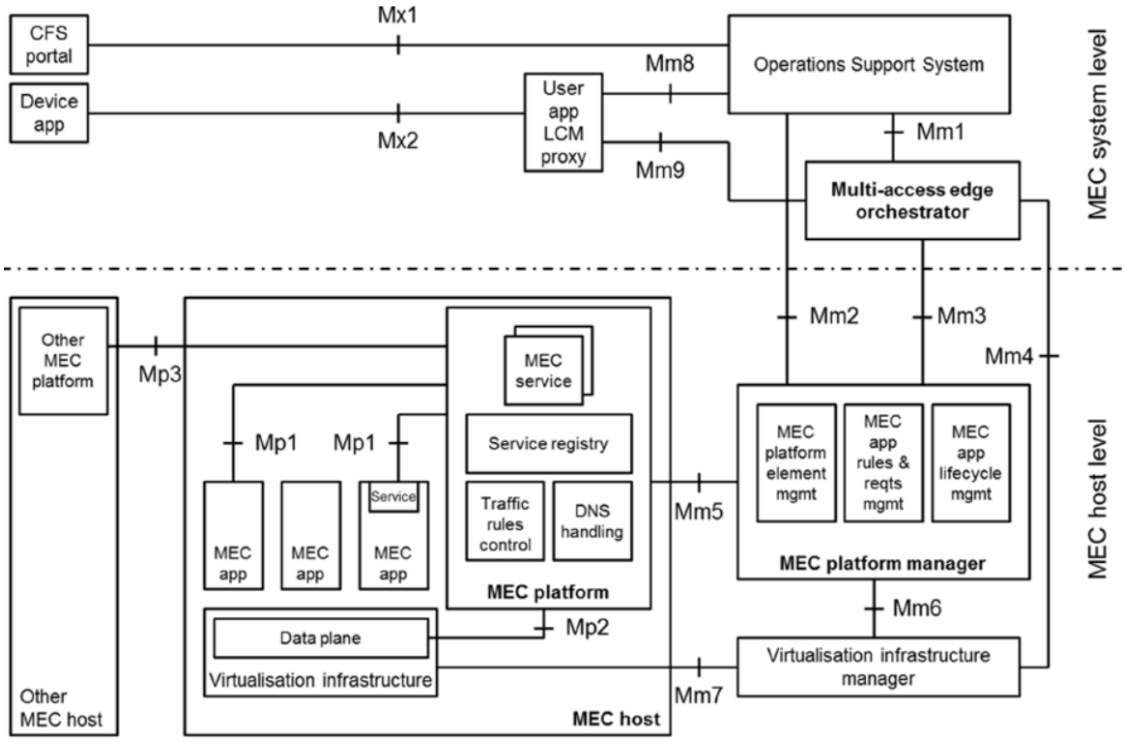

Figure 1: Multi-access edge system reference architecture [1]

Figure 2 illustrates how such MEC framework fits within the generic NFV reference architecture. In the proposed approach [2], both the MEC applications and the MEC platform are treated as Virtual Network Function (VNF)s, the Virtualisation Infrastructure as a NFV Infrastructure (NFVI) and the MEC platform manager and MEC orchestrator are replaced by the VNF managers and the NFV orchestrator, respectively.

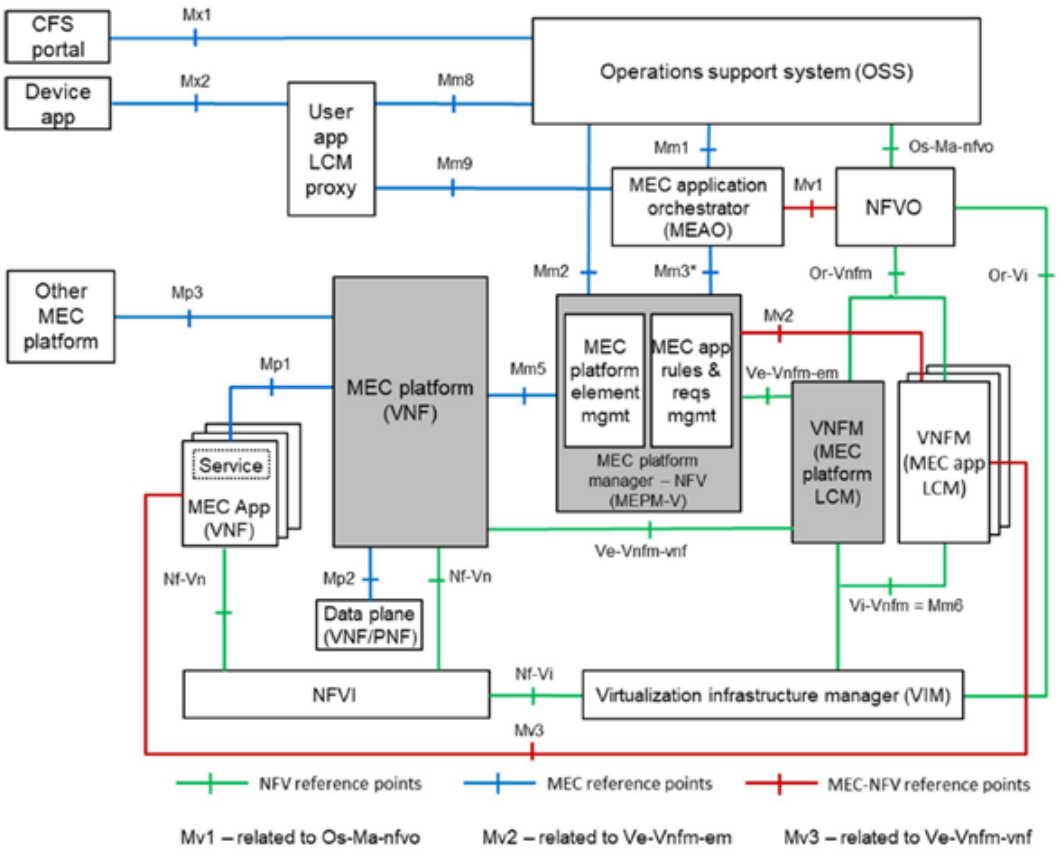

Figure 2: Multi-access edge system reference architecture variant for MEC in NFV [1]

Three main categories of use cases and applications have been already identified as part of the MEC specification: Consumer-oriented services (e.g., gaming, remote desktop applications, augmented and assisted reality, and cognitive assistance), Operator and third party services (e.g., active device location tracking, big data, security and safety, and enterprise services) and last but not least Network performance and QoE improvements (content/DNS caching, performance optimization or video optimization).

In the scope of CHARITY, these scenarios will be further researched to understand how they align with the envisioned next generation of distributed AR/VR and Holography-based applications. Relevant to CHARITY, two of the common MEC scenarios, discussed within the MEC standardization, are here highlighted.

The first, depicted in Figure 3, is the usage of MEC to support an AR service to cache and process the user location or camera view. This has the advantage of not only offloading such functionalities from the limited end-user devices but to also avoid deploying them into a remote cloud location, which would otherwise increase the latency and require the transfer of large amounts of data across all the network paths. This is especially important within densely populated scenarios (e.g., thousands of users in a stadium) where a high number of consumers are expected to see a particular content. Moreover, the data processing at edge can be also used to collect additional related KPIs which can then be leveraged to support service optimizations and improve the overall service QoE. The second scenario, depicted in Figure 4, refers to the leverage of the MEC server to deploy a video analytics application which is responsible for reporting the estimated network conditions (e.g., throughput available) back to the video server. Using such information, the video server can then adjust the transmission characteristics to maximize the capacity of the network. A full list of 35 scenarios considered can be found in the MEC specifications [3].

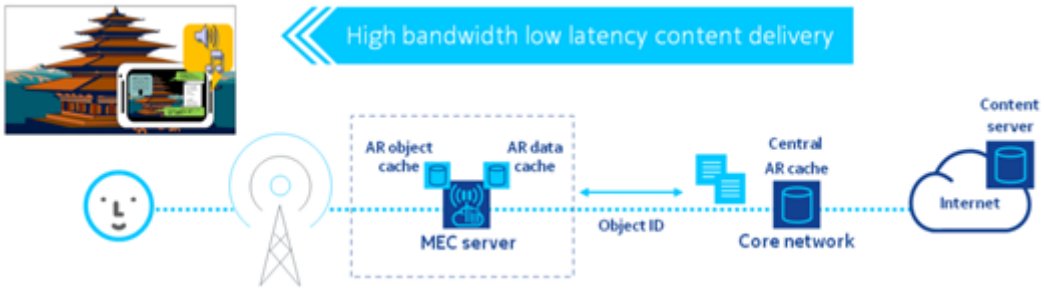

Figure 3: Augmented Reality Service Scenario [4]

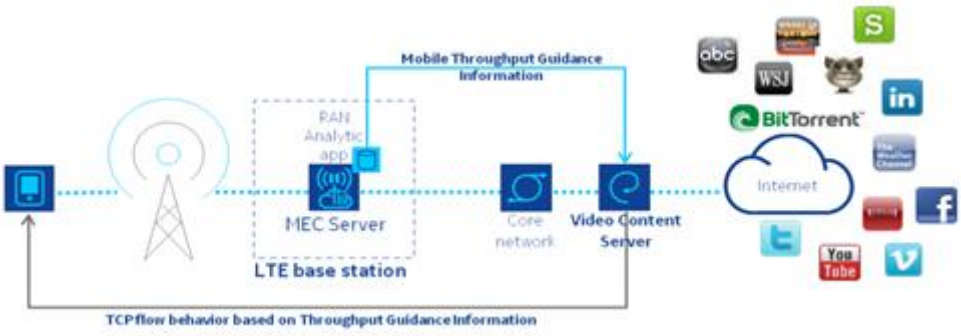

Figure 4: Intelligent Video Acceleration Service Scenario [4]

The deployment of applications across multiple administrative domains, one of the aspects considered by the CHARITY framework, is a challenge. Contreras, L. and Bernardos, C.[5] provided an overview of the integration options for MEC deployments across different administration domains (Figure 5:), including an integration at the infrastructure level, at the platform level, at the service level and the interconnection between two MEC systems.

Such comparison work helps to understand how multiple services (MEC applications in their case) can be integrated across such different deployments. It is also useful to understand the challenges of CHARITY when deploying the next-generation applications across multiple heterogeneous domains. They refer to the integration at the service level as the most straightforward option. Yet, regardless of the feasibility of all the options, the authors highlighted that technical and interoperability issues can complicate such deployments within such multi-domain environments, mentioning that several open challenges such as security, scalability, monitoring, accounting, or discovery automation needs to be further investigated.

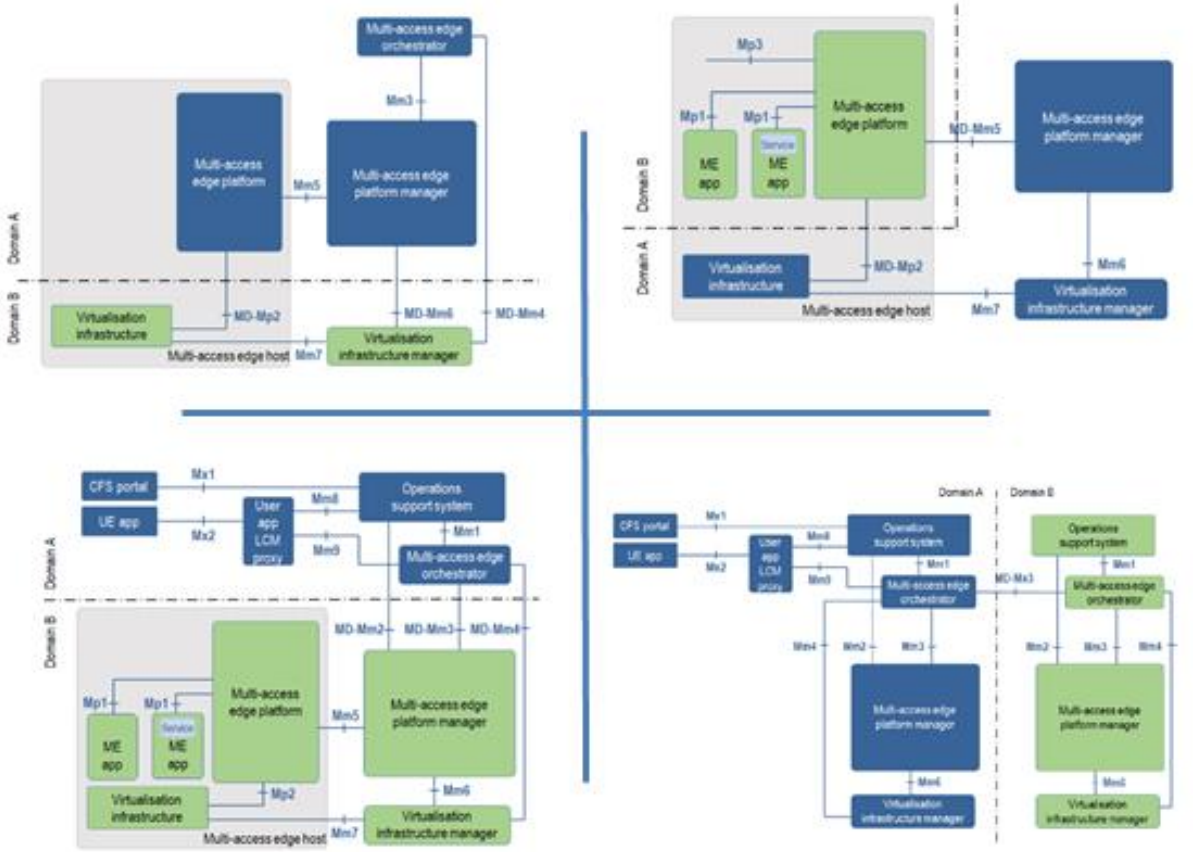

Figure 5: Different integration levels for MEC deployments: IaaS (top left), PaaS (top right), Service Level (bottom left), interconnection between two MEC systems (bottom right) – adapted from [5]

One of the recent ETSI MEC specifications [6] discusses the various integration patterns (Figure 6:) for connecting multiple MEC systems including scenarios involving MEC systems from different Mobile Network Operators (MNO) and the increasing relevant interconnection between MEC and cloud systems. That specification covers the high-level requirements of how each MEC platform discovers other platforms, how MEC platforms and applications can securely communicate with each other or even the application relocation across different MEC systems, a functionality envisioned by CHARITY.

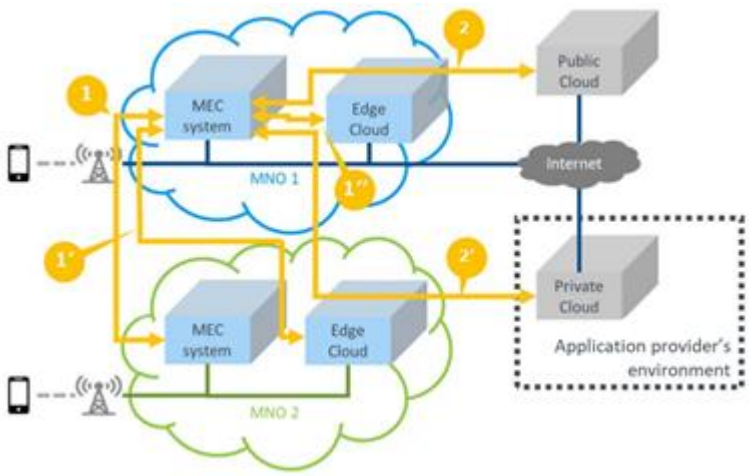

Figure 6: Patterns of relationship in Business and service layer [6]

Moreover, MEC specifications do also present an interesting MEC federation scenario for a location-based immersive AR game. As mentioned before, a MEC-based approach has manifold benefits for AR applications such as multiplayer gaming at a specific geographical area. Nevertheless, without integration between different MEC systems, such a scenario would be limited to players connected to the same MNO. In that specification, two solutions are proposed (Figure 7).

A first option, on the left of Figure 7, wherein each MEC application is responsible for creating an AR gaming room and both are responsible for the synchronization of gaming related components such as for users' game play actions, players' position, movement, direction, game control and the status of game contents virtually created. And a second option where one of the MEC applications is used to instantiate the gaming room and is responsible for serving the different users connected to different MNOs and transfer those details to the other application instance so the traffic can be redirected accordingly.

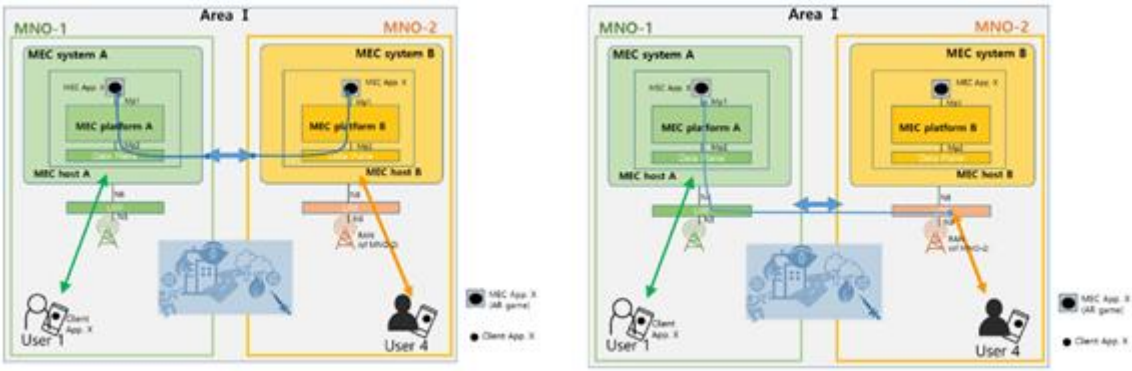

Figure 7: Two MEC federation options for a multiplayer interactive AR game scenario [6]

Additional exhaustive literature surveys regarding the manifold MEC possibilities, applications, and the role of MEC in 5G can be found in [7-9], whilst literature reviews targeting the usage of MEC for video streaming scenarios can be found in [10]. The latter discusses the MEC key technologies, resource allocation problematic, the optimization criteria, specifically focused video streaming scenarios such as AR/VR ones.

Relevance to CHARITY

Similar in spirit to MEC, CHARITY intends to leverage the benefits of deploying services (i.e., the next generation applications) across the edge/cloud and enabling service migration across MEC hosts/cloud domains - when needed. As presented before, numerous use cases can take advantage of edge components to, for instance, reduce the latency in network communications. The previous and extensive MEC work of identifying relevant use cases and architectural solutions is of vital importance to understand the requirements and challenges of these scenarios, i.e., how to decouple these use

cases into multiple services and how specific components can be leveraged to manage them. All of this work can be used to better understand, identify the components and tailor the CHARITY architecture to XR services. Moreover, the multi-domain problematic, a key topic already part of the MEC specification, can be seen as foundational research towards the service orchestration across different domains as part of Zero-Touch specification (Section 2.5) and the CHARITY architecture (Section 3).

2.2 ETSI Management and Orchestration

In 2012, fostered by the main players of the telecommunication market, ETSI created an ISG gathering more than 150 companies and research institutions, with the aim of creating the first standardized NFV reference architecture, and technical specifications for its specific components. This standardization effort is now at its Release 4, and has spawned a project providing an open reference implementation of its model, named Open Source MANO (OSM). In June 2021, OSM launched its Release 10.

From an architectural point of view, NFV specifications describe and specify virtualisation requirements, NFV architecture framework, functional components and their interfaces, as well as the protocols and the APIs for these interfaces. ISG NFV sets also specifications defining (in structure and format) how the deployment must be done, which features must be activated for VNF (this information are included in the deployment template), how all artefacts must be organized to be computable by the MANO framework. In addition, ISG NFV specification also covers security aspects related to virtualisation as well as performance, reliability and resiliency matters. The 5G emergence induced ISG NFV to approach new technical contents, such as multi-site and multi-domain deployments and network slicing, concepts that have been inspiring the design of the CHARITY architecture.

Related to new virtualisation technologies such as support for containerized VNFs (c-VNF) and container infrastructure management, in November 2020, ETSI has published its first specification enabling containerized VNFs to be managed in a NFV framework [11], namely ETSI GS NFV-IFA 040. This specification describes the new functions required for the management and orchestration of containers, the container infrastructure service management (CISM) responsible for maintaining the containerized workloads and manages the container, computation storage, network resources and their configuration, and the container image registry (CIR) responsible for storing and maintaining information of container software images.

The ETSI-MANO architectural framework identifies functional blocks and the main reference points between such blocks. The goal of the framework [12] is to obtain a high-level architecture where the software implementing the network functions is decoupled from the specific type of hardware and software used to manage the physical infrastructure. The decoupling exposes a new set of entities, the Virtualised Network Functions (VNFs), and a new set of relationships between them and the NFV Infrastructure (NFVI). VNFs can be chained with other VNFs and/or Physical Network Functions (PNFs) to realize a Network Service (NS). To do this, it is necessary to have an upper level of management and orchestration for the virtualised resources.

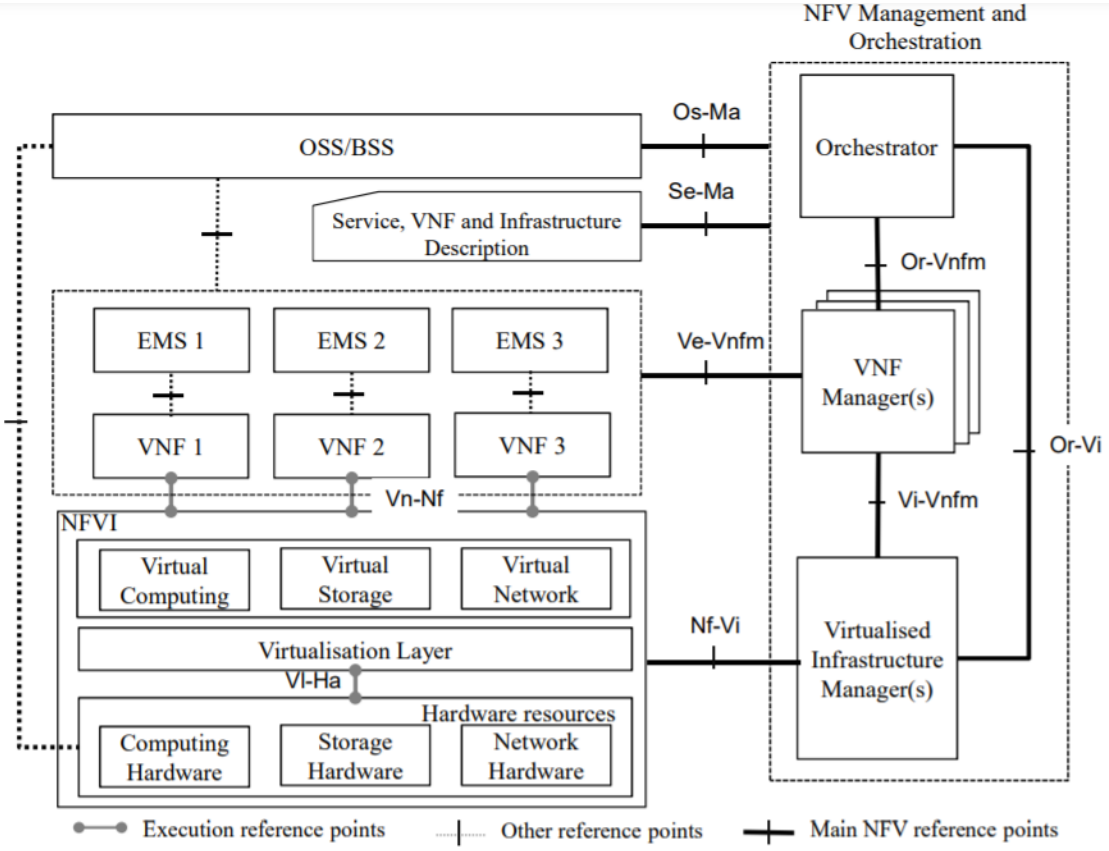

Figure 8: NFV Reference Architectural Framework [13]

Figure 8 represents the complete architectural framework defined by ETSI. The Network Functions Virtualisation Infrastructure (NFVI) represents the infrastructure of an NFV environment, providing hardware and software resources for the instantiation of network functions: CPUs, GPUs, memory, storage, network devices and software to virtualise resources (i.e., hypervisor-based or Container-based virtualisation). The NFVI infrastructure is designed to be distributed, so the NFVI nodes are decentralized in many locations (PoP's) supporting the locality and latency requirements necessary in some use cases. Inside this component, ETSI distinguishes three different domains: Compute Domain [14], Hypervisor Domain [15], and Infrastructure Network Domain [16]. In addition, it guarantees communication between VNFs and the Network Functions Virtualisation Orchestrator (NFVO), and between NFVI and NFVO. Virtualised Network Functions (VNFs) are the result of the NFVI layer virtualisation, implementing specific network functions like routers, load balancing, firewalls as software applications. Element Management (EM) performs the typical management functionality for one or several VNFs. It is responsible for FCAPS functions (*Fault*, *Configuration, Accounting* for the usage of NFV, collecting *Performance* measurement results for the functions provided by the VNF, *Security* management) for the VNFs. Operations Support Systems and Business Support Systems (OSS/BSS) provides essential business functions and applications such as operations support and billing. In a world that is quickly evolving, it is necessary for traditional OSS/BSS to adapt itself to new service flexibility, real-time service variations.

NFV Management and Orchestration (MANO) [17] is the component acting as coordinator/orchestrator of the whole NFV system. While the decoupling of VNFs from hardware shows many advantages in terms of flexibility, on the other hand, it carries out the need of a complex management layer that can handle different NFVs, different locations where they have to be instantiated in order to maintain appropriate service level, can allocate and scale in, out, up, down hardware resources to the VNFs, can manage the entire life cycle for each VNF, etc. These tasks (and many others) are referred to NFV MANO component. ETSI MANO provides methodologies to coordinate network service deployment operations and mechanisms for managing network functions. Some of the main functionalities offered by an NFV MANO module are as follows:

- NFVI virtualised resource management: availability, allocation and release;
- NFVI performance management;
- instantiation, resources scaling, updating, modification and termination of a VNF;
- instantiation, scaling (up or down), updating, modification and termination of a Network Service (NS);
- VNF or NS on-boarding.

In addition to these functions, there are many others that allow, for example, VNF and NS monitoring: through predefined metrics. These modules are able to evaluate when it is necessary to scale up or scale down the resources allocated to specific network service or network function.

ETSI MANO is composed by three main functional blocks: NFV Orchestrator (NFVO), VNF Manager (VNFM) and Virtual Infrastructure Manager (VIM). In addition to these, MANO architecture encloses also some data repositories like NS and VNF descriptor Catalogue, NFV Instances repository and all the resources needed to the VNF and to NS to be instantiated (*NS/VNF catalogue, NFV instances e NFVI resources*). NFV Orchestrator is the core element in the management of NFV architecture, in charge of NFV infrastructure orchestration and life cycle management of Network Services. In these responsibilities it is supported, at a lower level, by Resource Orchestrator (RO) for the first one and by Network Service Orchestrator (SO) for the second one at a higher level. RO manages the resources orchestration with the help of one or more Virtualised Infrastructure Manager (VIM): it decides their allocation in one or more NFVI-PoP (Point of Presence of NFVI, typically a node with resources located physically in the same point), and keeps track of the instances and resources that have been allocated for each single VNF in repositories depicted in Figure 9.

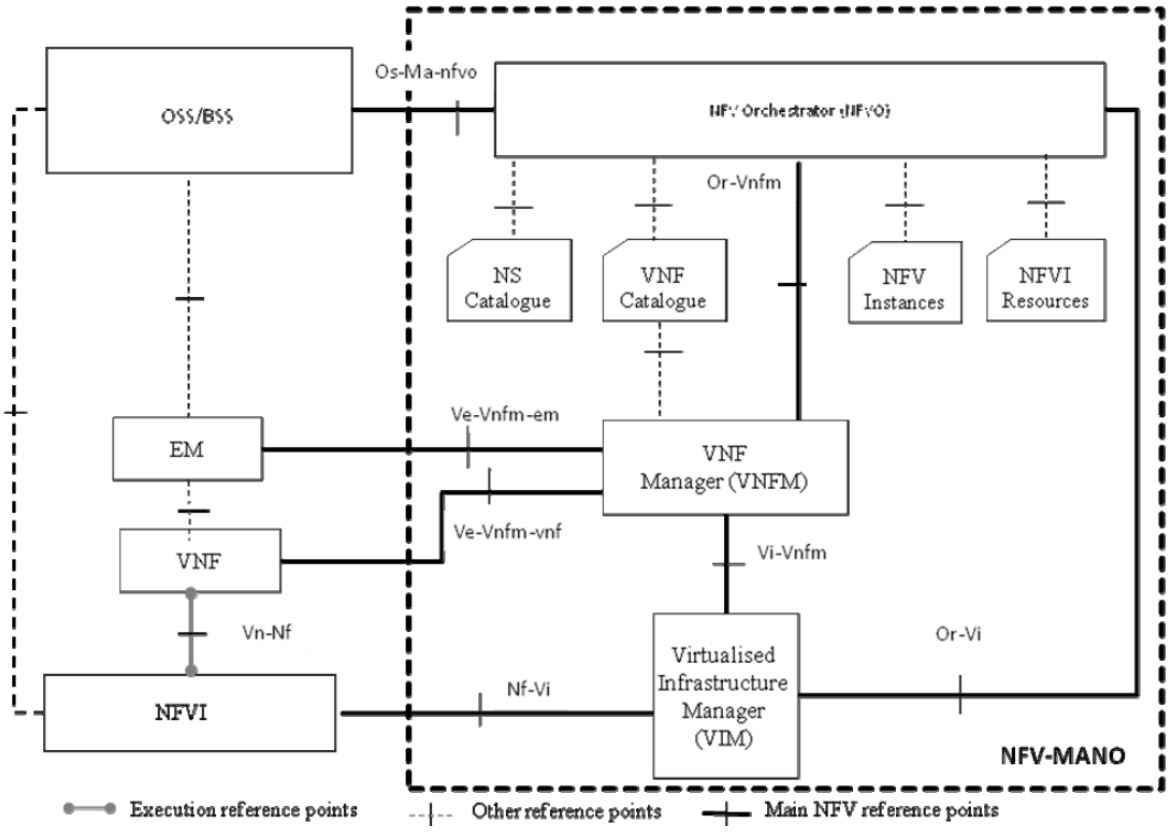

Figure 9: The NFV-MANO architectural framework with reference points [18]

The NSO works at a higher level, looking after the instantiation and management of Network Services as a whole, with their composing VNFs.

The VNFM handles the life cycle of the NFV or NS (instantiation, update, query, scaling, and termination). It can handle a single VNF or more VNFs, and often works with Element Management System (EM or EMS).

The VIM is dedicated to a high-level management of all (compute, storage, network) NFVI resources or only for certain type of NFVI resource (e.g. compute-only, storage-only, networking-only). It is responsible for low-level orchestrating the allocation (including the optimization of such resources usage) of the resources needed for the deployment of VNFs, keeping an inventory of virtualised resources mapping on physical resources. It also collects and notifies to the affected components information regarding infrastructure performance and malfunctions. In summary, VIM keeps an inventory of many kinds of resources, validates the requests coming from the Orchestration layer/module and executes them into the proper infrastructure layer.

Relevance to CHARITY

The CHARITY vision, as well as its conceptual architecture, has one of its foundational pillars in the NFV model and paradigm. The technical innovations provided by CHARITY are envisioned to be largely encapsulated and provisioned as VNFs and/or CNFs, and the blueprint of a CHARITY-enabled application is also expected to be essentially shaped as a Network Service composition, as described above. This makes the ETSI MANO architecture a key reference and potential baseline for the design of some core components of the CHARITY platform. Having a clear strong linkage to the NFV domain, the ETSI MANO is not only the first concrete model appeared on the NFV landscape, but also to date the most widely recognized and inspiring for multiple research initiatives. Thus, the CHARITY architecture has been designed with extreme attention to the recommendations and specifications coming from the ETSI MANO.

Among the elements potentially exploitable by the CHARITY architectural design (introduced in Section 3), we can consider:

- The structure and format of VNF Descriptors and NS Descriptors, as well as the breakup of VNF into single VNF Components, as starting point to define the CHARITY blueprint content;
- The structure and format of VNF Records and NS Records, as starting point for the runtime tracking of instantiated CHARITY resource pools;
- A number of components of the CHARITY architecture, in the XR Service Deployment Plane;
- The VIM specification as reference to define and design the analogous component in the CHARITY architecture, in charge of handling computational, storage and network resource pools;
- The WIM specification as reference to define and specify the analogous component in the CHARITY architecture, also in the XR Service Deployment Plane;
- The CISM functionality and services description provided by the release 4, as source to the CISM definition in CHARITY;
- The VNF Manager definition of general functionalities, source to the analogous component of CHARITY to handle the lifecycle management of VNFs;
- The NFVO (NFV Orchestrator) to define and design CHARITY NFVO component's functionalities of XR service onboarding, resource allocation and lifecycle management;
- The definition of interfaces between the above components, that CHARITY may seek to have as much as possible compliant with the ETSI MANO specification.

2.3 3GPP Edge Computing

Section 2.1 presented a general introduction to edge computing and discussed the reference architecture of Multi-access edge computing (MEC), a standardization initiative by ETSI ISG. In this

section, we will discuss the relevance of edge computing to 3GPP (Third Generation Partnership Project).

Similar to ETSI ISG MEC work, 5G networks based on 3GPP specifications leverage the network function interactions to align network virtualisation and SDN paradigms. In fact, ETSI has published a white paper [19] on how MEC can benefit the 3GPP ecosystem to enable application and service delivery at the edge of mobile networks. Figure 10 depicts the integrated MEC deployment by a 5G network. The MEC system comprises of MEC hosts and a MEC orchestrator, i.e., centralized system-level functional entity that can interact with Network Exposure Function or directly with a 5G NF. The distributed host functions are expected to interact with the 5G NF and are often deployed in the data network of the 5G system.

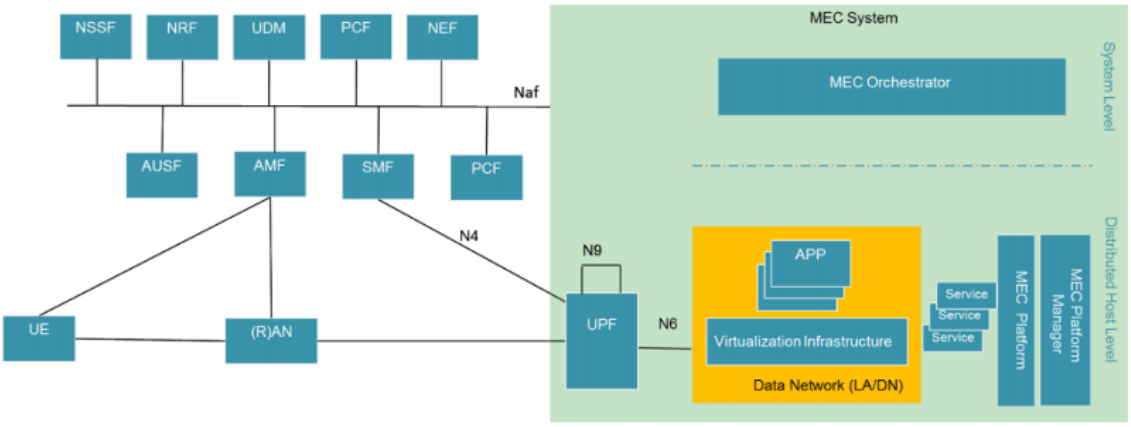

Figure 10: MEC deployment in 5G Network

MEC is deployed on the N6 reference point (as shown in Figure 10: MEC deployment in 5G Network), in a data network external to 5G system. The distributed MEC host is intended to accommodate MEC apps, a message broker and a service to steer traffic to local accelerators. The ETSI white paper mentioned above also describes various deployment scenarios where MEC hosts can be deployed in the edge or centrally and the UPF takes care of traffic steering to MEC applications. The four deployment scenarios include – (i) MEC and the UPF collocated with the Base Station, (ii) MEC collocated with a transmission node, possibly with a local UPF, (iii) MEC and the local UPF collocated with a network aggregation point, (iv) MEC collocated with the Core Network functions (in the same data centre), thus MECs can be flexibly deployed across any network components, from edge to a central data network.

As edge can be a huge benefit to the 5G, especially for latency sensitive applications, 3GPP has introduced edge computing as a priority area in Release-17 [20] aiming to provide native support for edge computing in 3GPP networks. The efforts cover application layer architecture [21,22], mobile core network enhancement [23], security aspects of edge computing in 5G core [24], multimedia streaming extensions [25] and management of Edge computing ecosystem [26].

Multiple stakeholders are involved in such edge deployments including

- Edge computing service provider (ECSP) – to build the infrastructure
- Mobile Network Operator (MNO) – to provide access to the infrastructure
- Application Service Providers (ASP) – to host application on the edge infrastructure

The architecture for enabling edge services has been presented in [21] (shown in Figure 11). The main objective of the work is to enable the communications between the applications running on the mobile device (Application Clients) and Edge servers (EAS) located at the network edge. The communication includes both control (service provisioning, edge discovery, mobility management between EAS) and data signals. The architectural principle includes (i) portability: Application logics of AC and EAS should not be modified while reaching the edge hosting services, (ii) differentiation: MNO should be able to

decide service differentiation, and (iii) flexible deployments. The architecture also paves the way to internetwork with existing 3GPP networks using existing capability functions such as NEF and PCF.

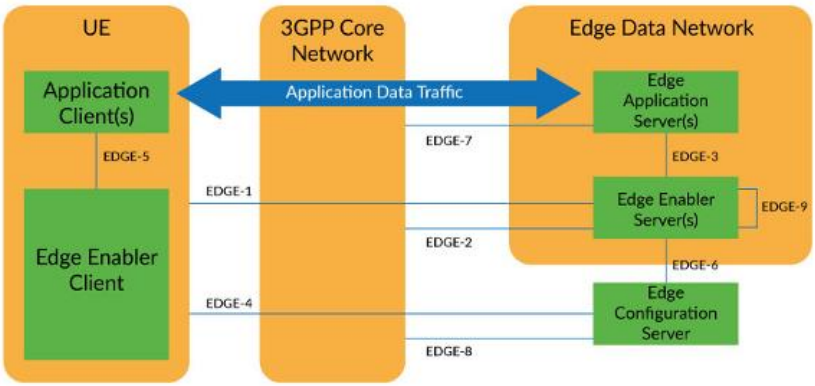

Figure 11: Enabling applications at the edge on 3GPP networks.

Thus, 3GPP's native support for edge computing aims to offer several capabilities which include – on-demand service provisioning, better availability of edge resources and resilience, network capability exposure via north bound API and support for edge/cloud continuum. The alignment of this architecture with ETSI to have a synergised Mobile Edge Cloud architecture has also been proposed [27].

Several open-source implementations are available to enable the deployment of edge computing solutions. The main ones include Central Office Re-architected as a Datacenter (CORD [28]), Low Latency Multi-access Edge Computing Platform for Software-Defined Mobile Network (LL-MEC [29]), and LightEdge [30].

Relevance to CHARITY

Communication between XR devices and other components in the XR service deployment plane (Section 3.3.2) is a vital part of the CHARITY architecture. We will adhere to 3GPP edge computing standards for defining the interfaces between the XR devices (UE) and the edge VNFs, especially when the access happens through a 3GPP (i.e., 4G/5G) cellular network.

2.4 Edge Storage

In edge computing, a large amount of data is generated and consumed by various edge applications. One of the key challenges in the development of applications at the edge is the efficient data sharing among the multiple edge clients. Data sharing can be realized within individual application frameworks or through an external storage service. In general, edge computing moves the computational load to the edge of the network in order to exploit the computational capabilities of edge nodes. Moving data and computation closer to the clients results in latency minimization and also improves network bandwidth. Thus, edge storage can greatly improve data access which in turn enables latency-sensitive applications. However, the distributed, dynamic and heterogeneous environment in the edge along with the diverse application's requirements poses several challenges.

Confais et al. [31] suggest a list of properties each edge storage system should employ. These properties are evaluated through a performance analysis on three off-the-shelf object storage solutions namely Rados, Cassandra and IPFS, using the Yahoo Cloud System Benchmark (YCSB). The performance is measured in terms of latency and network traffic. As the authors suggest, edge storage computing should address the following properties: low access time, network containment between "edge clouds" (cloudlets), availability through partitioning and mobility. Traditional file systems seem to be inappropriate solutions in edge environments due to their centralized nature. On the other hand, object storage systems which leverage Peer-To-Peer (P2P) mechanisms can cope better with the proposed edge storage requirements. Authors in [32] propose an epidemic approach for decentralized

storage systems which offer data publication, replication and retrieval. However, the proposed approach presents limitations regarding resource discovery. Another decentralized storage system for edge computing is presented by Gheorghe et al. [33]. The tasks and the data are submitted first on gateway nodes, thus leading to network latency minimization. For solving the network discovery problem, a network device discovery is employed which is able to find all the available devices for a certain application through the LAN router and forms a device tree for communication. Additionally, Sonbol et al. [34] propose a decentralized storage system called EdgeKV. It consists of a local layer with several groups formed by geographically "close" nodes and a global layer where different groups are connected through gateway nodes. These layers are connected through a ring overlay, which offers a fast and reliable system, utilizes data replication and provides strong consistency.

Furthermore, near real-time decisions can be efficiently improved by moving the analytics "close" to the data. As a result, edge architectures can reduce the amount of data traversing the network, thus minimizing latency and overall costs. Lujic et al. [35] propose a three-layer architecture model for data storage management on the edge. The edge layer manages data storage, performs local analytics, extracts information from available data and provides several components for monitoring and data preparation. In addition, as edge nodes do not have the same capacity as specialized storage devices, it is critical to determine which parts of data will be stored to improve real-time performance, while other less used data may be uploaded to the clouds. In order to address the problem of limited storage space in edge computing and to reduce data loss caused by unstable networks, authors in [36] propose a distributed multi-level storage system model which is based on a multiple-factors LFU (mLFU - Least Frequently Used) replacement algorithm. In general, replacement algorithms select parts of data to be removed from the current storage node when the storage space overflows. When the storage resource of an edge node is exhausted, part of data in the current node is chosen and uploaded to upper layers by the system through the mLFU. While direct algorithms like LFU are effective on the edge with restricted computational capabilities, they only consider access frequency of data. On the other hand, mLFU also considers the importance of data.

The edge of the network faces some challenges related to its decentralized management and security. By the integration of blockchain and edge computing many benefits can be obtained as stated in [37], such as reliability of the network and distribution of storage and computation over a large number of distributed edge nodes in a secure manner. Nonetheless, edge computing has several security issues related to control, data storage, computation and network. The migration of services across heterogeneous devices as well as edge servers may be proven risky. Furthermore, data integrity cannot be guaranteed, as many parts of the data are stored across different locations which would potentially lead to packet loss or inconsistency. Traditional methods for data detection result in heavy storage overhead in the edge storage. Maintaining security and privacy in computational tasks across a large number of computational nodes remains an open challenge. Thus, new solutions are required, capable of adapting to the decentralization, coordination, heterogeneity and mobility of edge computing. On the other hand, blockchain itself faces several challenges related to the massive storage required for transactions, the centralized risk of a large blockchain and throughput constraints. Although blockchain can guarantee a degree of security and privacy, outsourcing services at the edge are prone to transaction loss. Also, regarding resource management several challenges concerning the adaptation to the dynamic environments and the large-scale optimization for the collaboration of multiple edge servers must be addressed.

Caching at the edge can greatly improve data availability, retrieval robustness and delivery latency [38]. The performance of an edge caching algorithm is strongly related to the knowledge of content popularity among a number of users. Content popularity is the probability that a specific content item is requested [39]. However, the temporary content popularity changes dynamically over time. Therefore, machine-learning-based techniques are employed, in order to design efficient proactive caching algorithms. However, machine learning techniques are facing new challenges regarding data processing, limited computational resources, prohibitively expensive computational learning processes as well as privacy and security concerns. Thus, efficient learning schemes for massive high-dimensional data should be investigated and developed in order to provide accurate prediction of the

cached data at the network edge [40]. Zhang et al. [41] propose a learning offloading approach, which is able to decouple the high-complexity of AI-learning tasks and assign them to distribute edge computing resources. Edge-to-edge cooperation is achieved, thus increasing the efficiency of learning due to its high QoS and utilization of idle learning resources. Finally, Ale et al. [42] propose an online proactive caching model based on a bidirectional recurrent neural network (BRNN) which is able to predict the time-series user requests.

The guiding principle in CHARITY is to provide a distributed storage spread across heterogeneous edge and cloud nodes, with intelligent decisions on data placement and considerations on performance and security. Thus, CHARITY is responsible for delivering optimized edge storage services to the CHARITY framework and its hosted applications. These services include data storage, retrieval tasks, security and privacy, QoE insurance and mitigation as well as other data related services that serve the runtime requirements.

Relevance to CHARITY

As traditional LFU algorithms employed on the edge consider only the access frequency of the data, CHARITY will borrow ideas for the evaluation of data "importance" from various fields including fault tolerant distributed data stores, thus globally improving the storage's hit rate. Furthermore, CHARITY will provide optimized data prefetching through intelligent admission mechanisms which are able to identify the correct time frame that data should be prefetched to the edge, preserving them as cache. When considering the "correct time", the network bandwidth and activity as well as the resource availability of the involved edge devices and possibly their mobility and reliability will be taken into consideration. Machine learning and predictive analytics mechanisms will be employed, aiming at a more concrete prediction model, optimizing the off-loading process by preventing bottlenecks and violations on QoS and QoE expectations of the platform. Furthermore, as caching can greatly improve data availability, retrieval robustness, and delivery latency, several efficient machine learning-based schemes such as deep learning, supervised/unsupervised learning, reinforcement learning and transfer learning will be utilized in order to provide a new way of edge caching development. Finally, when it comes to security, the most popularly investigated option is the use of blockchains. The intention is to deal with issues such as reliability of the network and distribution of storage and computation over a large number of distributed edge nodes in a secure manner. The integration of blockchain and edge computing would bring some benefits in terms of security, privacy and automatic resource usage. Thus, a Distributed Hash Table (DHT) in the blockchain will be employed which is able to overcome data integrity issues by turning the blockchain into an automated access-control manager.

2.5 ETSI Zero touch network & Service Management (ZSM)

5G technology and network slicing are reshaping the classical approach of how the services and infrastructure were orchestrated in the past. Service, telecommunication, and infrastructure providers are now looking into more flexible ways of having fully automated and E2E service management which might span under the umbrella of distinct domains, both administrative and technological. To address that problematic, ISG ZSM from ETSI was formed to discuss relevant use cases, requirements, and specify an E2E management reference architecture that allows such end-to-end service deployments.

Figure 12 shows the ZSM framework reference architecture proposed by ETSI [43]. This architecture is conceptually composed by six building blocks: Management Services, Management Functions, Management Domains, E2E Service Management Domain, Integration Fabric and finally, Data Services.

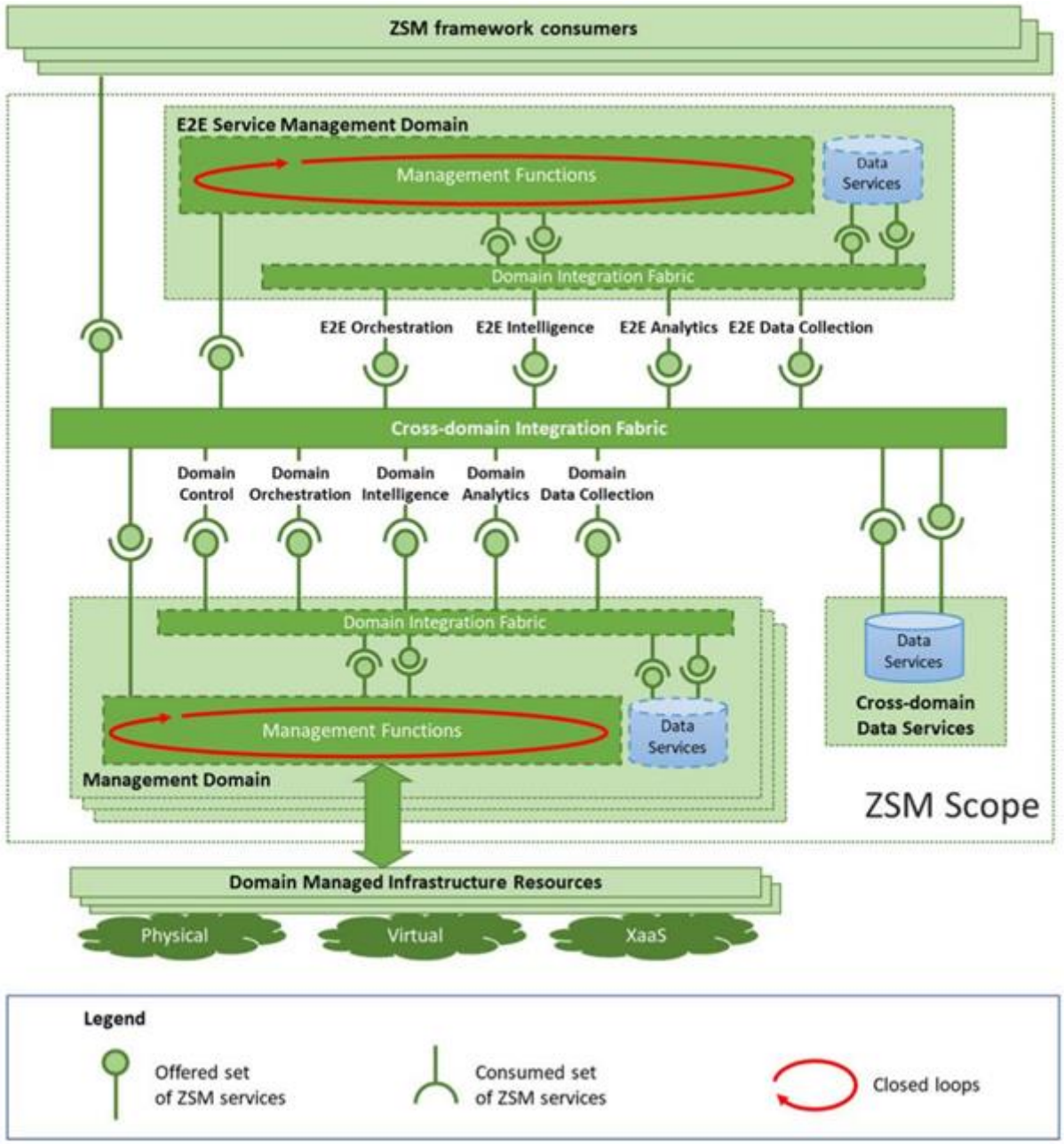

Figure 12: ZSM framework reference architecture [43]

ZSM Management Services, exposed through specific endpoints, allow a more consistent and standardized way to expose different management capabilities across the overall deployment. This is an important factor given the discussed scenarios comprise services spanning over different domains. ZSM Service capabilities are offered (produced) and/or used (consumed) by Management Functions. Multiple capabilities can be combined to form broader abstractions of management features. Indeed, collaborative, and federated service orchestration models were also considered in ZSM, playing a relevant role in scenarios involving different operators.

Such Management Services are then organized by functionality into Management Domains. Within each domain, there can be internal or exposed services, depending on whether their access restricted to domain or exposed to outside of the domain. Like service capabilities, the ZSM specification also considers the possibility of hierarchy at the Management Domains where multiple domains being recursively stacked on the top of each other.

Moreover, the ZSM framework specifies an E2E Service Management Domain. Among others, the E2E Service Management domain is responsible by end-to-end orchestration across different domains, E2E closed loop management, E2E analytics, and data collection.

Another pivotal building block proposed in the ZSM framework is the concept Integration Fabrics. They are meant to facilitate the communication between management functions. Indeed, ZSM defines two types of Integration Fabrics: Domain Integration Fabric and the Cross-Domain Integration Fabric. The first one is responsible for connecting services within the same domain. While the second is used to facilitate the communication over distinct domains. Note that such fabrics are not only used as a communication bus between services (which shall support for synchronous and asynchronous communications) but to facilitate the registration, discovery, and invocation of the different supported services. Finally, ZSM also comprehend the concept of Data Services which allows to decouple and to reuse the same management data across distinct management services.

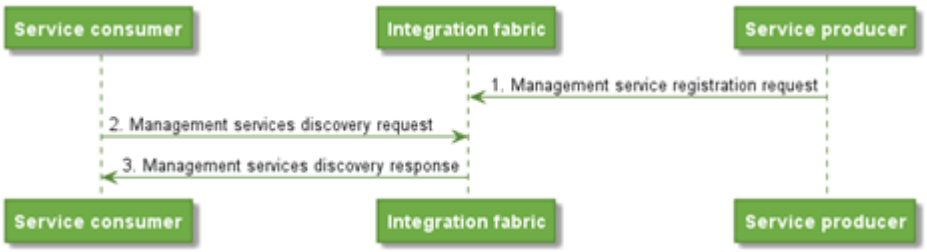

Figure 13: ZSM Service Registration and discovery [43]

Figure 13 and Figure 14 illustrate some of the communication patterns referred in ZSM specification regarding the service discovery and registration as well different communication possibilities that ZSM framework shall support (i.e., publish/subscribe approach and synchronous communications).

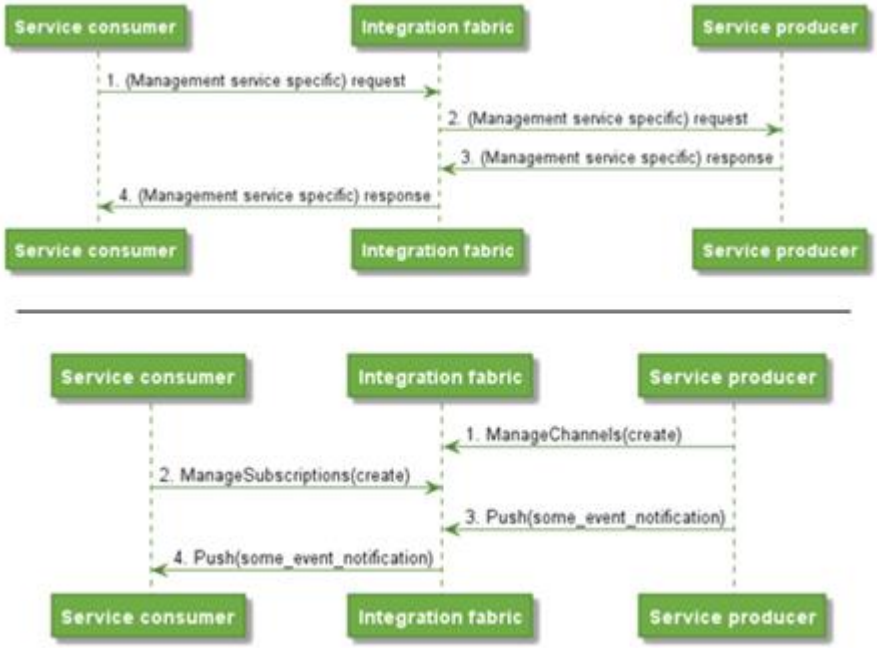

Figure 14: Synchronous Request-response (top) vs Pub/Sub via push (bottom) [43]

The ZSM framework was conceived based on a set of principles such as modularity, extensibility, scalability, model-driven, open interfaces, closed-loop management automation, support for stateless management functions, resilience, separation of concerns in management, service composability, intent-based interfaces, functional abstraction, and simplicity. Together, these principles allow to obtain a future-proof design architecture that can be used to fully automate (i.e. Zero-Touch concept) the network and service management, which, again, can span across multiple domains, both

administrative and technological. Likewise, the ZSM specification also defines a set of requirements that shall be satisfied by a given ZSM implementation. They range from non-functional requirements such as the need to be vendor, operator, and service provider agnostic to functional requirements such as the proposed support for adaptive closed control loops or the E2E and cross domain support. Likewise, within the same specification, ISG ZSM defined a set of security specific requirements covering aspects such as the need for confidentiality and integrity of management data at rest, in transit and in use. Those requirements were based on a set of related use cases scenarios as described in [44]. The complete list of requirements can be found in the ZSM Reference Architecture specifications.

More than E2E deployments, the underlying idea of ZSM is to achieve a level of automation where closed-loop processes and algorithms (e.g., machine learning based orchestration mechanisms) can drive more efficient and flexible scenarios (e.g., a self-monitoring and optimization of the network) and ultimately reduce (or eliminate) the need for human intervention. Indeed, the concept of Closed Loops, which can occur at both Management Domain and E2E Domain levels Figure 15, is further discussed into an additional ETSI specification [45].

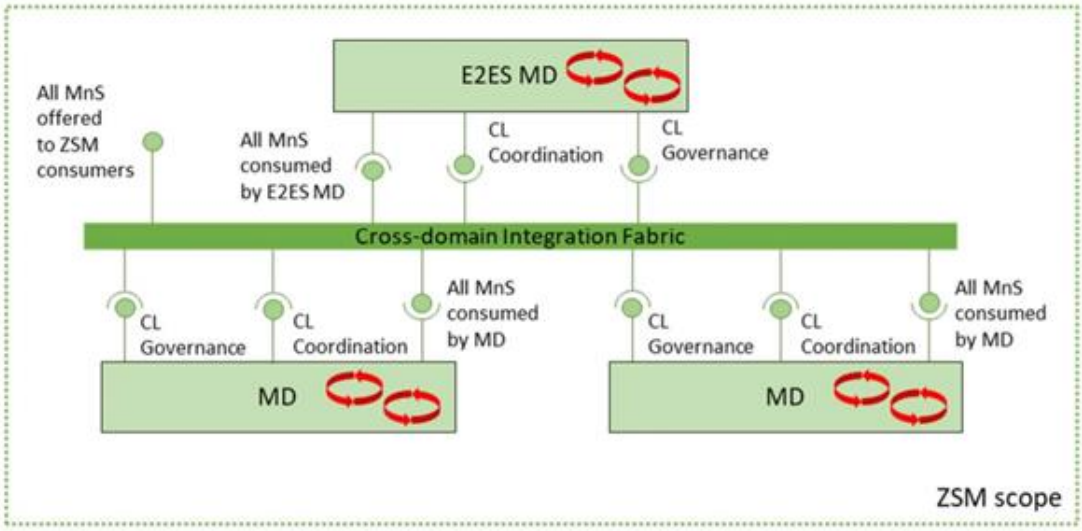

Figure 15: Closed Loop and related management capabilities [45]

Each of those loops are logically split into a set of ordered stages accordingly to some of the well-known closed loop approaches (e.g., OODA loop (Observe, Orient, Decide, Act) or MAPE-K (Monitor, Analyse, Plan, Execute)). Figure 16 provides an example of how closed loop stages map into a generic example of a network fault. In such an example, data is collected from the network (Observe), whenever an abnormal condition is detected, it escalates the issue for an analysis (Orient), then a subsequent evaluation decides what should be done (decide) and finally a solution is applied to fix the issue (Act).

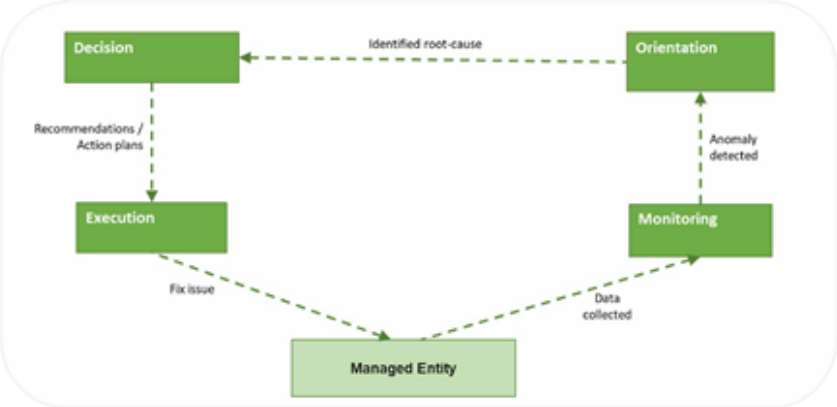

Figure 16: Example of Closed Loop stages in a Network Fault scenario [45]

Relevance to CHARITY

The ZSM specification is of utmost importance towards the standardization of how applications and services spanning across multiple domains can be fully realized. Such specification includes the definition of how common orchestration functions could be implemented and how the different components can communicate. Inspired by that, CHARITY framework was conceived to achieve such notion of E2E orchestration across multiple domains which will be used to support the deployment of next-generation applications (i.e., the XR services) within a cloud native environment. CHARITY leverages the Integration Fabrics concept to connect all the components of the CHARITY framework both within each domain and across domains. Those integration fabrics, as detailed in Section 3, are pivotal to facilitate the service discovery, registry and all the service orchestration related functions. They allow a more decoupled, efficient and scalable way to connect different components. Moreover, CHARITY also leverages the notion of closed loops as a structured approach to enable the deployment of both service-related orchestration functions and XR specific mechanisms (e.g., adaptive video streaming, to allow applications to adjust to network conditions based on the metrics collected in the real-time or to implement a given security function).

2.6 Experiential Network Intelligence — ENI

During the last decade, there have been numerous instances where the implementation of Artificial Intelligence in real-word applications has provided ground-breaking results. In order for networks to leverage the functionalities enabled by the use of Artificial Intelligence methodologies the European Telecommunication Standards Institutes introduced the Experiential Network Intelligence (ENI) framework. The purpose of this framework is to provide guarantees in regards to the network's ability to keep up with the Quality-of-Service requirements. More specifically ENI is able to assist or direct network management systems based on network status and Service Level Agreements.

The ENI entity is in charge of providing recommendations or commands to an Assisted System (AS), in order for intelligent network management to be established. It is possible for the ENI to communicate with the Assisted Systems via an Application Programming Interface (API) broker that performs the appropriate translations. Up to this point, three classes of AS have been identified based on the degree that the various Artificial Intelligence mechanisms influence the management and orchestration of the network.

The backbone of the ENI architecture is the implementation of closed control loops. These closed control loops are based on the Observe-Orient-Decide-Act (OODA) paradigm. Furthermore, there are two distinct types of control loops. The inner control loops, each one of which consists of multiple loops that enable the basic OODA functionalities to be conducted in parallel. The outer control loops that guarantee that the overall OODA process is carried out in accordance to a desired output state.

The various phases of the OODA loop process are implemented by utilizing some predefined functional blocks. The "Observe" phase is the product of the Input Processing and Normalization functional blocks. This structure is in charge of cleansing, curating and combining the various heterogeneous data inputs. Furthermore, the Processing and Normalization functional block consists of the Data Ingestion and the Normalization functional blocks. The purpose of these blocks is to handle the various data formats and then alter them in a way which is compliant with the format of the entities that are expected to receive them.

The "Orient" phase of the OODA loop is based on the use of the Knowledge Management, Context-Aware Management, Cognition and Situational Awareness functional blocks. The Knowledge Management functional block is able to facilitate Machine Learning and formal logic by providing formal representation of the ingested data. The Context-Aware Management functional block monitors closely the changes in the state of the environment in order to provide adaptability. The Cognition functional block is responsible for establishing high-level goals in regards to optimizations and Service Level Agreements, which are either learned or provided by the operator. Finally, the

Situational Awareness functional block examines the events taking place and evaluates the potential impact of various available actions on the future state of the Assisted System.

The "Decide" phase of the OODA process relies on the Model-Driven Engineering and Policy Management functional blocks. The former takes a set of suggested changes and translates them into a set of actions. The latter turns these actions into reusable policies.

The "Act" phase is carried out by utilizing the Denormalization and Output Generation functional blocks. These blocks are in charge of producing a denormalized output which can be digested by the various Assisted Systems. In some cases, the API broker may be required to perform additional translations before the output gets transferred to the appropriate Assisted System.

In the case of XR applications, it is of major importance to be able to provide carrier grade assurance capabilities in order to ensure that the QoS requirements will be met. Capabilities such as these heavily rely on the need to conduct optimal resource allocation between competing network slices. The QoS requirements are provided by the customers in the form of Service Level Agreements. The various SLAs are then transferred through the OSS/BSS Assisted System to the Orchestration and Management Assisted System and the ENI system. The SLA requirements are then processed by the Data Ingestion and Normalization functional blocks. During the same time-frame, the Infrastructure Assisted System establishes and configures the corresponding network slices according to the blueprints created by the OSS/BSS Assisted System. This process leads to the normal operational phase of the ENI System, during which the Data Ingestion and Normalization functional blocks gathers raw data from the environment. The normal operational phase is interrupted when an abnormality in regards to the expected resource consumption occurs. The Situational Awareness functional block is responsible for storing the associated configurations up to the point that the abnormality occurred. Then, it is up to the Situational Awareness and the Model-Driven Engineering functional blocks to operate together in order to prevent any potential violations of the SLAs from taking place. The Situational Awareness functional block is in charge of deciding which one of the suggested actions will be taken. After the appropriate plan has been decided, it is up to the Model-Driven Engineering functional block to translate it into a set of commands that can be interpreted by the various network components. The role of the Policy Management functional block is to translate the action plan into a set of policies, which are then forwarded to the Denormalization and Output Generation functional blocks. These blocks translate the policies into a format which can be understood by the Assisted Systems.

Relevance to CHARITY

CHARITY aims to leverage the ENI paradigm in order to automate network (and XR service provisioning) optimization. There are two main driving forces behind the process of network optimization in regards to orchestrating the available resources. The first one is the limited computational capacity of the existing nodes and the second one is the cost of acquiring additional resources. This process of horizontal auto-scaling provides additional self-healing capabilities in the case of node failures. Since the process of acquiring additional resources requires a time-span which may be up to several minutes, it is necessary to proactively allocate before a bottleneck occurs. A Traffic Prediction mechanism will be providing information regarding the resource utilization that is expected to take place in the near future. This information shall enable the orchestrator module to proactively allocate computational and network resources according to the produced prediction. In addition to that, a Deep Reinforcement Learning mechanism shall be utilized in order for the proper actions to be performed at each specified time-step as far as the allocation/deallocation of resources is concerned. Furthermore, a similar approach will be taken in regards to routing and scheduling traffic flows. Based on the results provided by the Traffic Prediction mechanism, the upper bound of latency provided by the developers and the current network topology, a Deep Reinforcement Learning mechanism will be in charge of performing traffic steering in a manner which guarantees that the QoS requirements will be met. Both these functionalities are designed in a manner which is compliant with the ENI paradigm. Further details on these mechanisms will be provided in D2.1 of WP2, as the relevant work are not within the scope of WP1.

2.7 OpenRAN

The evolution of cellular networks, especially, in the last two decades has facilitated better and seamless connectivity to the Internet. Various technological innovations and milestones have been achieved over various generations of cellular access and the most recent fifth generation of cellular networks has not just seen upgrades in terms of network availability and speed but also in terms of openness, transparency and security.

Open radio access network is one such initiative to move towards open radio networks from a conventional closed network, by supporting interoperation between networking devices from various vendors. The three primary elements in the RAN includes (i) Radio Unit (RU), where the radio signals are processed, (ii) Distributed Unit (DU) comprising of real-time baseband functions, and (iii) Centralized Unit (CU) where less time-sensitive packet processing functions reside. The OpenRAN defines standard interfaces to communicate between these RAN elements breaking the siloed nature of traditional RAN, and at the same time, to bring down the costs of deployment for network operators. Moreover, open RAN standards are developed using vRAN principles, offering security, flexibility at a diminished cost.

3GPP plays a major role in facilitating open RAN, especially for 5G, through splitting the gNB (5G radio base station) into a CU and DU, as shown in Figure 17. The CU can be implemented as a Virtual Network Function and DU as a Physical Network Function, with the communication between CU and DU [46].

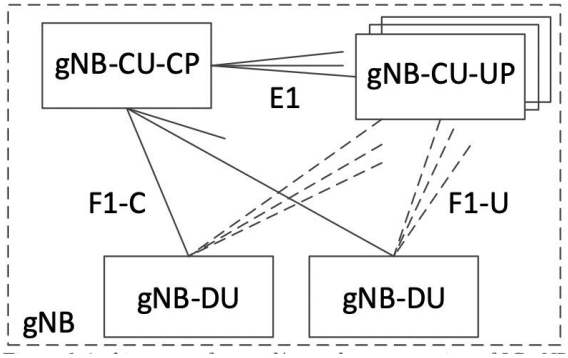

Figure 17: Architecture of control/user plane separation of 5G gNB

Several industrial initiatives are extending the standards beyond the 3GPP specifications mainly due to the expansion in industrial IoT devices. There are three main such industrial alliances that facilitate to have the open ran. This includes –

O-RAN Alliance

The O-RAN alliance was founded to define requirements and help build a supply chain ecosystem to realize openness and intelligence for evolving radio access networks. Openness allows RAN implementations to scale and enable smaller vendors/operators to introduce customized services, building a competitive ecosystem. On the other hand, intelligence would help in automating the deployments and operating the network in an automated fashion. O-RAN alliance was formed by operators and later became a community of vendors, and research & academic institutions.

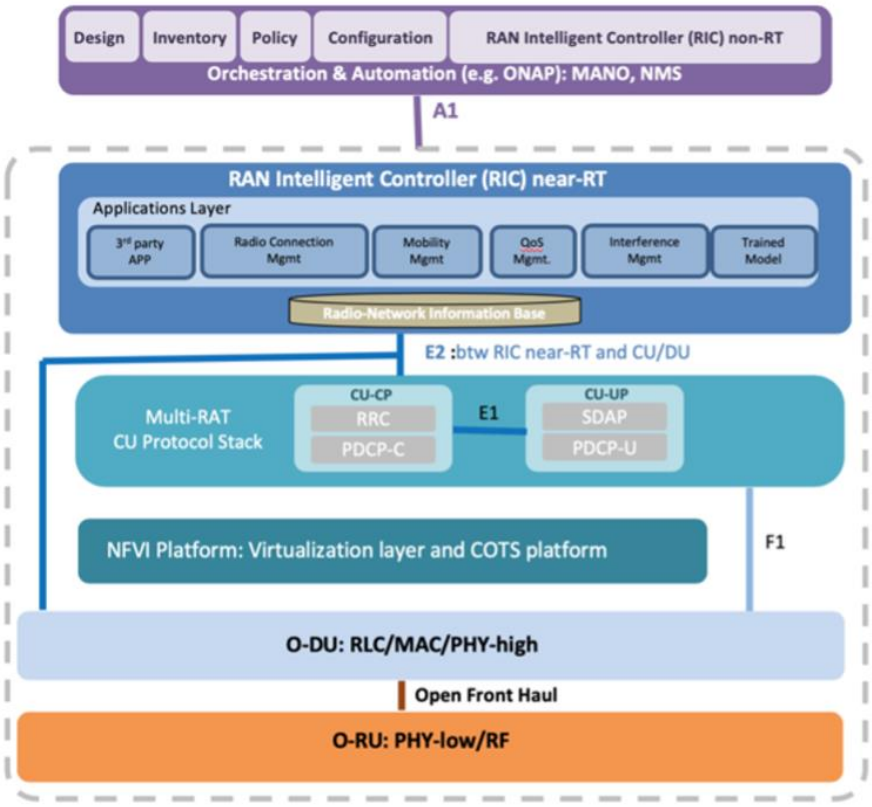

Figure 18: Reference architecture of O-RAN alliance[2]

A key principle of the O-RAN architecture is to extend SDN concept of decoupling the Control Plane (CP) from the User Plane (UP) into RAN alongside embedded intelligence. This extends the CP/UP split of Centralized Unit,

- being developed within 3GPP through the E1 interface (Figure 17), and
- further enhances the traditional radio resource management functions (assigning, reassigning, and release radio resources in single/multi-cell environments) with embedded intelligence.

The intelligence is introduced through the hierarchical (Non-Real Time -RT- and Near-RT) RAN Intelligent Controller (RIC) with the A1 and E2 interfaces (Figure 18). The Non-RT function (i.e., >1s) include service and policy management, RAN analytics and model training; the trained models and real-time control functions are distributed to the RIC for (near) real-time executions. Ninkam et al. [47] show the benefits of implementing Open RAN through specifications from O-RAN alliance through evaluation of real-world cellular data. The work also points the challenges and open problems in the area.

OpenRAN

OpenRAN project group was announced by TIP (Telecom Infra Project), a consortium of hundreds of service and technology providers, to enable open architecture design and flexible deployment of RAN equipment. OpenRAN aims in building disaggregated RAN functionality through open interface specifications, whereas O-RAN alliance is a specification group defining next generation RAN infrastructures.

The key philosophy behind OpenRAN includes disaggregation of RAN HW & SW on vendor neutral platforms, open interfaces to enable interoperability within vendors, flexibility in terms of choosing the cellular-generation, hardware agnostic deployments and innovation via adopting smartness – all intended to take a holistic approach towards building next-generation RAN.

[2] https://www.o-ran.org

OpenRAN architecture has been split into four components and two segment subgroups. The component subgroups look into the open/disaggregated hardware implementations of Radio Unit and Distributed/Centralized Unit and as well on the Radio Intelligence and OpenRAN orchestration and life-cycle management. The segment subgroups investigate the outdoor macro deployments and indoor small-cell deployments.

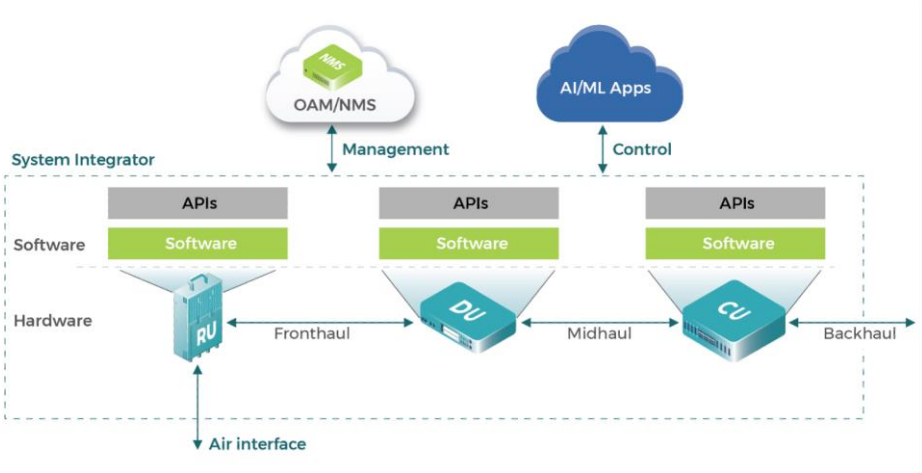

Figure 19: Reference architecture for TIP-OpenRAN

The RAN architecture is split into two parts. *(i)* the lower layer RAN split between the antenna integrated Radio Unit (RU) and the Distributed Unit (DU). *(ii)* higher layer RAN split, a 3GPP standard F1 interface between the DU and the Centralized Unit (CU). The F1 interface is expected to comply with 3GPP F1 interface specifications (Figure 19). The management system of the OpenRAN provides interfaces that comply with 3GPP Integration Reference Points (IRP) specifications. The system is expected to provide management, configuration, monitoring, optimization and troubleshooting capabilities. The OpenRAN and O-RAN alliance have also signed a liaison agreement to ensure alignment in developing interoperable RAN solutions [48].

Open RAN policy Coalition

Open RAN policy Coalition (ORPC) is a more recent group aimed to advocate for government policies to support the development and adoption of open radio access network. In a sense, the coalition aims to bring a policy-perspective to complement the standardization and technical implementations done by O-RAN and OpenRAN communities.

Overall, the goals of ORPC include supporting the O-RAN and OpenRAN communities, calling the government to support for open and interoperable solutions, creating policies that support vendor-diversity, and enabling funding and promoting open radio deployments [49].

Furthermore, operators and vendors have published their views and commitments to open RAN through these alliances (Nokia [50], Telefonica [51], NTT Docomo [52], Ericsson [53]). Several research work and frameworks to enrich the capabilities of RAN have also been made open, following the CU/DU split specifications from the alliances. Some of them include O-RAN [54], COMAC [55] and SD-RAN [56].

Relevance to CHARITY

OpenRAN exposes RAN-relevant intelligence that can be certainly leveraged by the CHARITY architecture to improve the service provisioning of XR services to mobile users. For example, OpenRAN can be explored to schedule user requests at RAN to enable a seamless and faster access by the users while taking into account the features of the provisioned XR service. How CHARITY can optimally explore OpenRAN interfaces to support XR services is yet a research problem that needs further investigation.

2.8 New IP and Deterministic Networking

During the last decade, there has been a dramatic change in the very fabric of the applications produced. Concepts such as Holography and Cross Reality are no longer considered novelties but actual application features that need to be implemented in a manner that guarantees that the various QoS requirements are met. The fact that it is of paramount importance for these applications to be able to operate in real time in order to provide an immersive experience to the end users, makes them extremely latency – sensitive. On top of that, Holography – based applications in particular requires by nature huge amounts of bandwidth to be allocated. In other words, it is imperative to introduce network mechanisms that are able to provide i) guaranteed low end-to-end latency, and ii) nearly optimal utilization of the available bandwidth. This necessity has led to the creation of the New IP initiative which was introduced by the ITU-T Network 2030 Focus group. It is the study of various technologies that have been identified as of vital importance for the next evolution of the Internet. These technologies aim to provide advanced flexibility and deterministic services in the already-established network paradigms. In order to do so, it is essential to re-examine the modus operandi of certain aspects of the Internet data plane, the protocols involved and the subsequent architecture. More specifically, the New IP initiative entails two key concepts that need to be taken into consideration in order to establish efficient forms of networking that are able to support next-generation applications.

The first one is the ability to support Multipath Routing. Multipath Routing is the simultaneous management and utilization of multiple paths in order to transmit streams of data flows. The implementation of this concept offers certain benefits such as fault tolerance and increased bandwidth. Each stream is assigned a separate path. By doing so, multiple transmission queues are created, thus ensuring better utilization of the available bandwidth. In case the number of streams exceeds the number of available paths, some of them are chosen to share the available paths. On top of better transmission performance, Multipath Routing introduces advanced fault tolerance by assigning an alternative path to the stream, should the established one fail.

One way of implementing Multipath Routing is by utilizing routing strategies that operate in combination with existing protocols. Equal-Cost Multipath Routing is the most notable Multipath Routing strategy. It is implemented by distributing traffic among various equal cost paths by utilizing hash functions. One significant drawback of using this routing strategy is that it does not take into consideration the changes that are bound to occur to the network status. This inability to keep up with the dynamic nature of the network leads to sub-optimal load balancing. In [3], the Internet Engineering Task Force introduces Multipath TCP which is a set of extensions to standard TCP. These extensions enable transport connections to take place by utilizing multiple paths simultaneously. However, all the solutions explored to this point fail to provide low upper bound of end-to-end latency. In [57], Latency-controlled End-to-End Aggregation Protocol (LEAP) is introduced. LEAP is a multipath transport layer protocol that provides probabilistic end-to-end performance guarantees thanks to path multiplexing and inter-flow coding. The utilization of packet-level encoding enables the information of each data flow to be carried through all the available paths. The information is then retrieved at the destination. In [58], a rather similar approach is explored in the context of UDP for video streaming.

The second key concept presented by the New IP initiative is the ability to facilitate deterministic services. The significance of this characteristic becomes rather apparent when contemplating two distinct requirements that the majority XR applications share. The first one is the low end-to-end latency. In addition to that, some XR traffic flows are temporally correlated to other ones. This fact introduces the necessity to facilitate synchronization among specific traffic flows. In order to handle these issues, the Deterministic Networking paradigm[4] was established. Although it is clearly stated that

[3] https://datatracker.ietf.org/wg/mptcp/documents/

[4] https://datatracker.ietf.org/wg/detnet/about/

Deterministic Networking is not involved with the modification of transport protocols, like the variations mentioned above, the study of technologies that operate alongside them is well within its scope of operations. The cornerstone of Deterministic Networking is the creation of deterministic data paths that are able to guarantee bounds of latency, loss and jitter. The data paths are formulated in the context of layer 2 bridged and layer 3 routed segments. The New IP initiative focuses on large layer 3 networks and more specifically on developing methods of flow identification and packet forwarding over layer 3. By classifying data flows, it is possible to establish specific data paths for the time-sensitive traffic flows. The reservation of specific network assets for each latency-sensitive service provides network-layer certainty of information transmission. Furthermore, by classifying data flows, it becomes possible to simultaneously facilitate time-sensitive and best-effort services. This takes place by distributing the available transmission mediums between DetNet and non-DetNet flows in a fair manner. DetNet-enabled devices contain ports, each of which is equipped with a specific number of queues that are utilized by DiffServ and Best-Effort traffic. Each DetNet-enabled device in the network can be configured to utilize per-class or per-flow queuing.

The IEEE standard named IEEE 802.1Qch[5] (Cyclic Queuing and Forwarding) introduced the concept of cyclic operations in regards to coordinating queue and dequeue functionalities. The utilization of CQF relies on dividing time into intervals. Two queues are utilized for each class in order to perform enqueue and dequeue functionalities in separate time intervals. That means a traffic flow that arrives in interval x shall be put in the first queue and shall be transmitted via the second queue during the (x+1) interval. The resulting frame then shall arrive to a specified switch during the same time interval. As one can see this sets a harsh bound that dictates that the propagation latency has to be less than the selected time cycle. As a result, CQF is not suitable for large scale networks due to the inherit difficulty of properly choosing a suitable time cycle. In [59], Cycle Specified Queuing and Forwarding is proposed in order to solve this issue. CSQF is rather similar to CQF with the exception of utilizing explicit description of the various transmission cycles at every DetNet node present in the path spanning from source to destination.

Relevance to CHARITY

CHARITY aims to leverage the DetNet paradigm, as well as get inspired from the design of New IP, in order to provide guarantees regarding the network's ability to keep up with the established QoS requirements. Towards this goal, it is essential for a data flow template to be established. This template shall entail information regarding the source, the destination and the desired upper bound of latency. There are two distinct classes of time-sensitive flows that CHARITY aims to facilitate. The first class is indicative of data flows that require extremely low end-to-end latency in the form of the aforementioned upper bounds of latency. This requirement is associated with time-sensitive data flows and more specifically with real-time applications. The second one is associated with the need to properly facilitate the various temporal correlations which are established among some specific data flows. The upper bound of latency which is provided by the next-gen developer will be regarded as a time-stamp which dictates the exact moment that each traffic flow has to arrive at its perspective destination node. The ability of each DetNet flow to arrive at its destination node at a specific moment is guaranteed via the use of Cyclic Queuing and Forwarding protocols like the ones which were examined above. The priority of each traffic flow is established based on its latency-sensitive it is. This type flow prioritization will be leveraged by a Deep Reinforcement mechanism which will be in charge of formulating optimal routing strategies in accordance to the upper bounds of latency. Furthermore, a Traffic Predictions mechanism will be providing estimates regarding the traffic which is expected to take place in the near future. Alongside the information provided by the data flow template, it is essential to offer information about the topology of the network in order to implement centralized configurations in regards to routing and scheduling. These configurations are established via the SDN paradigm which is capable of configuring schedules on the hosts and the forwarding tables of the

[5] https://1.ieee802.org/tsn/802-1qch/

switches. Further details on these mechanisms will be provided in D2.1 of WP2, as the relevant protocol-level work are not within the scope of WP1.

2.9 XR-Relevant Standards

2.9.1 3GPP

3GPP (3rd Generation Partnership Project) is a worldwide consortium of organizations focused on development and maintenance of mobile standards. Since 1998, they have been responsible of GSM, LTE, UMTS and many other protocols. Their main objective is shared by CHARITY, ensure the compatibility in the heterogeneity of the network.

The consortium organizes their work in periodical releases (Figure 20). Release 15 was the last complete document published. Between the different projects developed, there is one topic addressed frequently in WP1 of CHARITY Project, 5G End-to-End Network Slicing. This property of the 5G Systems allows to use multiple types of services and apply them to different network requirements (latency, priority, type of users, etc.) at the same time. This feature is not only important for the design of the CHARITY architecture, it also offers important features for the interoperability between operators and service providers, that can tailor their communications according to the needs of the networks and limit the resources per slice.

Release 16 is complete but still in production due the worldwide difficulties caused by the COVID-19 pandemic. It will go deeper in the 5G system thanks to different use cases and needs born through the Release 15 investigation, applying the mobile standard to all the spectrum of mobility, from the space to the sea.

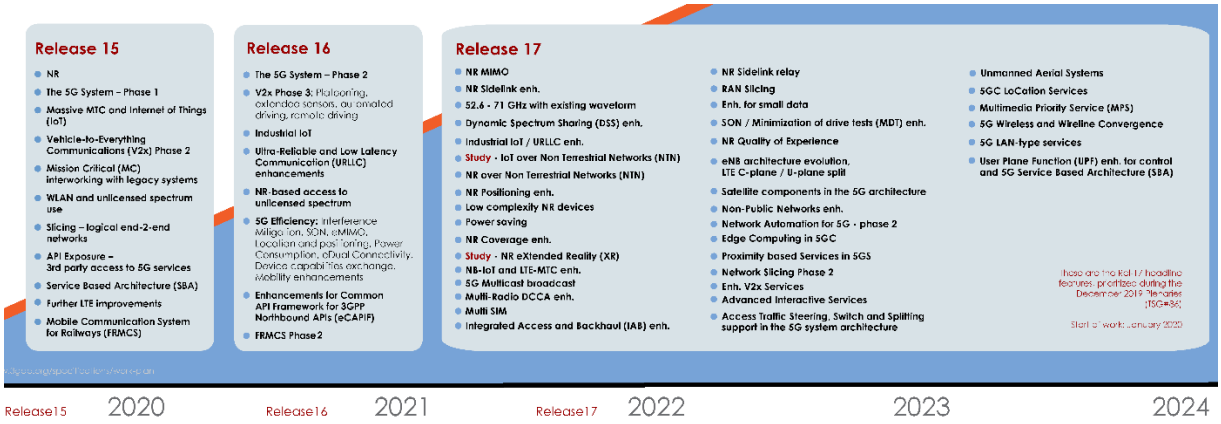

Figure 20: Headline features of 3GPP releases[6]

The studies of Release 17 began in January 2020 and are also affected by the pandemic. It covers highly specific topics and the Extended Reality has found its place in the agenda as in the industry. Their first step is evaluating the performance of XR in terms of power consumption, capacity, mobility and coverage, in partnership with all the top companies of the mobile industry. As in the CHARITY Project, they have distinguished three main fields of use cases: virtual reality, extended reality and cloud gaming. The specificity of the study is not only in the physical devices and their resources, but also in the spaces where the applications tend to take place. The numerous companies attached to the study, from Facebook to Xiaomi, confirm the high interest of the industry in the possibilities of the Extended Reality.

[6] https://www.3gpp.org/ftp/Information/presentations/presentations_2020/Poster_2020_MWC_v6_OPTIMIZED.pdf

2.9.2 3GPP MEDIA STREAMING

The focus of mobile networks has long shifted from analogue to digital and from voice to data. The predominant challenge of mobile networks is how to continuously move more data and move it faster than before. The official entrance of 5G to the mobile stage began with Release 15 and included many advances and innovative designs to achieve bigger bandwidths, lower latencies and more connected devices than ever before. The breadth and ambition of 5G is enormous – from the re-architecting of the core as a Service Based Architecture, to Multi-Access Edge computing, from Networking Slicing to a huge expansion of the radio access network. Within the context of CHARITY, of particular interest is the specification work carried out by 3GPP in the domain of media streaming over mobile networks. XR has often been cited as one of the drivers for 5G requirements [60] and as a highly challenging vertical that demands both bandwidth increases and latency decreases.

Our examination of 3GPP media streaming focuses on Release 16 – the latest official release at the time of writing – and the delivery of time-continuous media [61]. The 3GPP work tackles a somewhat different problem to that faced by CHARITY in that it focuses on streaming media to a user who has no interactive role in composing that media [62,63]. Its model centres on a one-way download and thus envisages frame buffering (on the client and potentially at the edge) as a core component. There are, however, a number of interesting characteristics and design attributes in the 3GPP architecture that can prove useful for CHARITY.

2.9.2.1 Control and Data Separation

Throughout the 3GPP 5G specifications from the radio access network to the core, we observe a clear separation between the control and data planes and this is again evident in the 5G Media Streaming Architecture. In Figure 21 (adapted from [60]), we see a high-level view of the 3GPP architecture for downlink media streaming. We show, for informational purposes, the typical deployment sites for the various components. The 5GMS client typically runs on the user device and is used by third party applications running on the device to interact with the operator media streaming infrastructure.

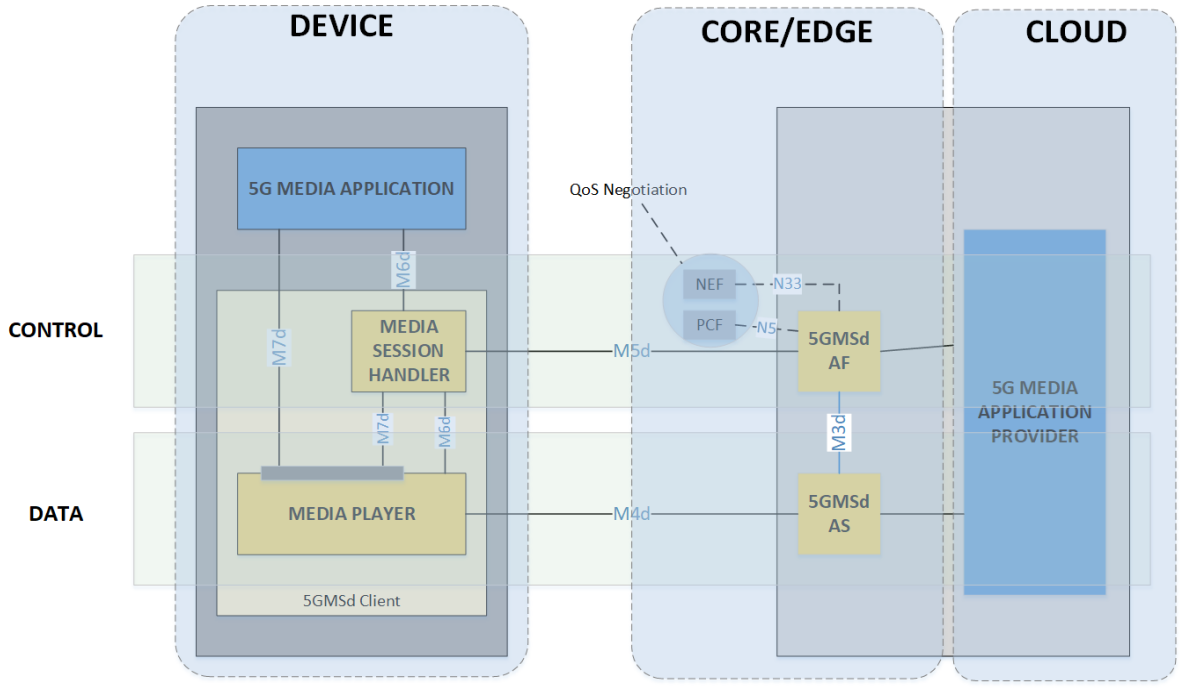

Figure 21: Control and Data Plane separation in 5G Media Streaming

The Media Application Provider typically installs their server-side application on a public cloud and connects it to the 5G data and control planes through standardised APIs to stream their media to 5GMSd-Aware applications. The Media Server takes responsibility of the data plane – decoding,

decryption, presentation, monitoring – while the Control Plane is managed by the Session Handler whose purpose is to establish, control and support the delivery of a media session.

2.9.2.2 QoE Support and Monitoring

Depending on the level of trust agreement with the operator, server side applications will have access (through Network Exposure Function -NEF- of 5G Core) to functions within the operator network to assist in QoE targets. Release 16 witnessed the introduction of Downlink Network Assistance to the media streaming architecture. This enables a user device that is receiving a downlink media stream to improve the session QoE. As well as clients being able to measure the downlink traffic rate, they can now ask the network what the most appropriate bit rate currently is or what it will most likely be for in the next nominal period. In addition, if a client is running out of buffered content, then it can request a temporary delivery boost from the network. This feature may also be used at the beginning of a session for a faster bootstrap experience.

The instrumentation, gathering and reporting of metrics and consumption figures is an area that 3GPP invested some considerable attention in and the design they arrived at is quite powerful. While the media server component is heavily instrumented to be able to gather useful metrics, it does so only at the behest of the session handling infrastructure. Metrics configuration is done on the network level, for instance defining which geographical areas that shall have metrics collection active, which metrics to collect, and how metrics shall be reported. Metrics are periodically reported to AF which periodically reports to the central server – possibly after carrying out some aggregation and filtering.

2.9.2.3 CHARITY support of 3GPP Media Streaming

While CHARITY heeds and leverages the architectural decisions made in the design of the 3GPP Media Streaming architecture, it also supports its realization. CHARITY seeks to be deployable on heterogeneous cloud data networks – whether on the public clouds of hyperscalers or private clouds of enterprises – and optimize media delivery either directly to the 5G User Plane Function or to the specialized 5G Media Streaming downlink (5GMSd) Application Server.

Furthermore, CHARITY will extend its reach to the 5G Multi-Access Edge and manage the edge-cloud continuum for downlink media streaming so that 5G User Equipment such as a phone equipped with the 3GPP 5GMSd Client can connect to the nearest 5GMSd Application Server within the operator network which in turn is served by an application hosted in the CHARITY media streaming infrastructure spanning the edge and cloud.

2.9.3 MPEG Mixed & Augmented Reality

The MAR Standard (Mixed and Augmented Reality) is a reference model established to define required modules, minimal functionalities and the associated information content and models for applications, components, systems, services that must claim compliance with MAR systems. This reference model was published as ISO/IEC 18039 as a technical report defining the scope and key concepts of MAR including the relevant terms and their definitions and a generalized system architecture. The MAR-RM is agnostic to platforms, used devices and algorithms. It does not specify how MAR applications should be designed, developed and implemented. The objective of the MAR reference model is to establish the definitions, main concepts and overview of the architecture needed to create mixed and augmented reality systems or applications. The reference model is planned for use by current and future engineers of MAR applications, parts, frameworks, administrations, or particulars to depict, analyse, contrast, and impart their compositional plan and execution.

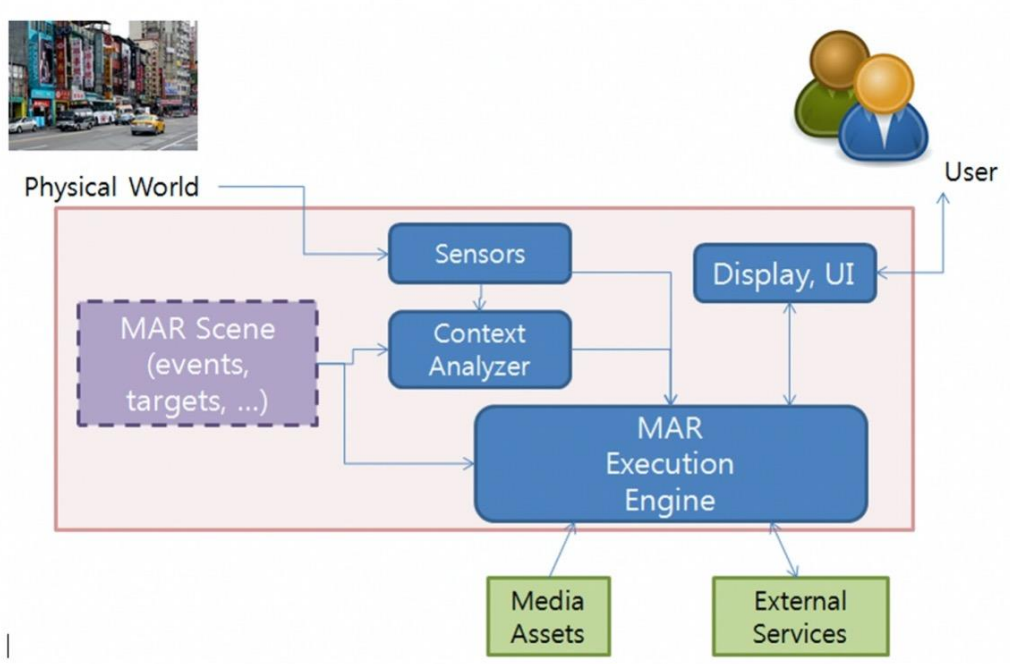

Figure 22: MAR Reference System Architecture [64]

Table 2 depicts the reference system architecture of the MAR standard. After information is being sensed from the physical world, it is delivered in the MAR execution engine either directly or through the context analyser. The MAR execution engine may also access media assets and required external services. The engine processes the sensed information, outputs the processing outcome, and manages user interactions with the system.

MAR reference model viewpoints

The Enterprise viewpoint (Figure 23) verbalizes the perspective of the business elements in the framework that ought to be justifiable by all partners. This intentionally focuses on scope and policies while it also presents the targets of various actors involved. It defines the actors involved in a MARS (Mixed and Augmented Reality System), the associated potential business models and the desirable characteristics at both ends of the value chain.

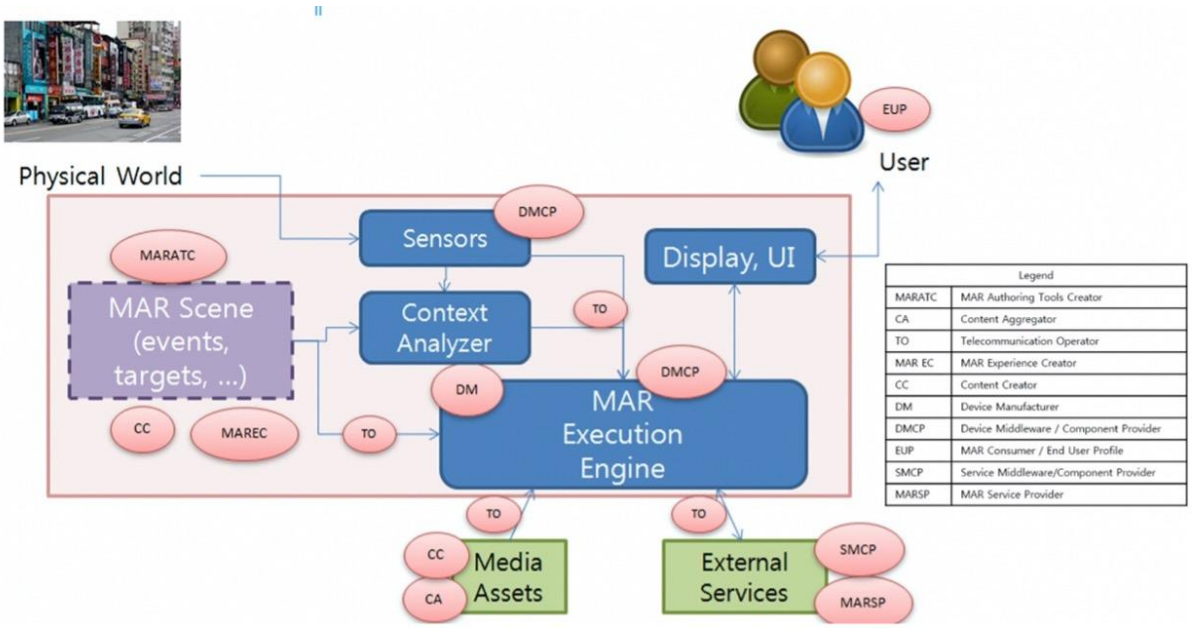

Figure 23: MAR Enterprise viewpoint [64]

The actors may be categorized in four classes:

a. Providers of Authoring/Publishing: MAR Authoring Tools Creator, MAR experience creator, and Content Creator.

b. Providers of MAR execution engine components: Device manufacturer and Device middleware/component Provider.

c. Service Providers: MAR Service Provider, Content Aggregator, Telecommunication Operator, and Service Middleware/Component Provider.

d. MAR User: MAR Consumer/End-User Profile.

The Computational viewpoint identifies the main processing components (Figure 24), both hardware and software, and define their roles and interconnectivity [65].

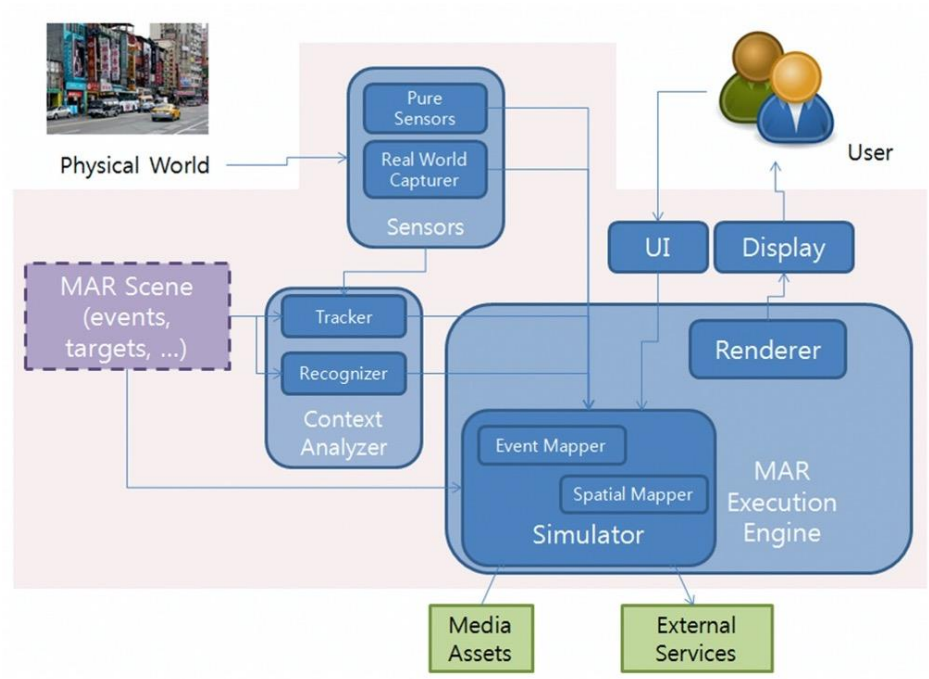

Figure 24: MAR computational viewpoint

Table 1: MAR Components [66]

Component	Implementation	Roles
Sensor	hardware	A sensor is a device used to detect, recognize, and track the target physical object to be augmented. Captures properties-measurements of the physical world. It interprets and converts them into digital signals. This observed data may be utilized by tracker/recognizer to evaluate the context and/or to compose the scene. There is a variety of types of devices (cameras, environmental sensors, etc.) that measure different physical properties.
Context analyser - Recognizer	Hardware / software	The recognizer as part of the context analyser, consists of two systems: is a system that analyses signals sensed from the physical world by conducting comparisons with local or remote target signal (i.e., target for augmentation). It produces MAR events and data.
Context analyser -	Hardware /	The Tracker, as part of the context analyser, aims to

Tracker	software	detect and measure changes in the target signals physical properties (e.g., pose, orientation, volume, etc.).
Execution engine	software	Its main objective is to further recognize and track the target to be augmented by interpreting the sensed data. It imports the object data from the physical world. It also creates computational simulations of the dynamic behaviour of the augmented world. Finally, it integrates the physical and virtual data before rendering within the required modalities (e.g. visuals, aurals, haptics).
Simulator -Spatial mapper	software	The spatial mapper, as part of the simulator within the execution engine, computes spatial relationships (position, orientation, scale, and unit) between the physical world and the MAR scene by applying the required calibrating transformations. Furthermore, it maps each sensor's spatial reference frames and spatial metrics.
Simulator - Event mapper	software	The event mapper, as part of the simulator within the execution engine, is responsible of associating MAR events, obtained from the Recognizer or the Tracker, with conditions specified by the MAR Content creator in the MAR scene.
Renderer	Software / hardware	The Renderer, as the part of the execution engine, produces the output signal for the presentation of the MAR scene simulation. It is responsible of converting the MAR scene into the proper form of output signal for the given display device.
Display-UI	hardware	The display device presents actual MAR scene to the end-user in various modalities. Displays and UI include monitors, head-mounted displays, projectors, scent diffusers, haptic devices, and sound speakers.

MAR system classes

- MAR Class V: augments the 2D visual data captured by real camera.
- MAR Class R: augments the 2D visual data continuously captured by one or multiple real cameras and reconstructs the 3D environment (note: SLAM, PTAM).
- MAR Class A: augments the aural modality.
- MAR Class H: augments the haptic modality.
- MAR Class G: uses global position system to register synthetic objects in the real world.
- MAR Class 3DV: augments the 3D visual data captured by multiple cameras, video, and depth cameras.
- MAR Class 3DA: augments the scene by using 3D audio data.

The MPEG-MAR XR standard will drive the design of the CHARITY platform to suit accordingly the AR (and possibly Holographic) use cases. It will be further investigated whether CHARITY may contribute an extension to this XR standard in the direction of AR streaming services over 5G cellular systems.

2.9.4 Omnidirectional MediA Format [OMAF]

Immersive media technologies such as virtual Reality (VR) combine omnidirectional media and head-set displays to provide users with an immersive experience. Omnidirectional media includes videos that have been captured using 360-degree cameras with a field of view that covers approximately the entire sphere in the horizontal plane.

Omnidirectional MediA Format (OMAF) is a virtual reality (VR) system standard developed by the Moving Picture Experts Group (MPEG). OMAF enables omnidirectional media applications - 360° video, images, audio and timed text (text media synchronised with other media). At the time of writing this deliverable, OMAF v2 fully supports three degrees of freedom (3DOF), while support for six degrees of freedom (6DOF) is still progressing. This will allow for transitional user movement to prompt the rendering of overlays and for multiple viewpoints.

Requirements for OMAF v3 include support for new visual volumetric media types, such as video-based point cloud compression (V-PCC) and immersive video. The MPEG standard for visual volumetric video-based coding and V-PCC has been finalized and can be used to represent captured volumetric objects. The MPEG Immersive Video standard was planned for completion in July 2021 and enables 6DOF within a limited viewing volume. It is expected that the OMAF standardization for integrating these media types will start later this year.

Figure 25 presents the OMAF architecture, showing the three main modules; (1) OMAF content authoring module, (2) delivery access module, and (3) the OMAF player module.

- OMAF content authoring module - media acquisition, omnidirectional video/image preprocessing, media encoding, and media file and segment encapsulation.
- OMAF delivery access module - may either use file delivery or streaming delivery for which the content is time-wise partitioned into segments.
- OMAF player module - media file and segment decapsulation, media decoding, and media rendering.

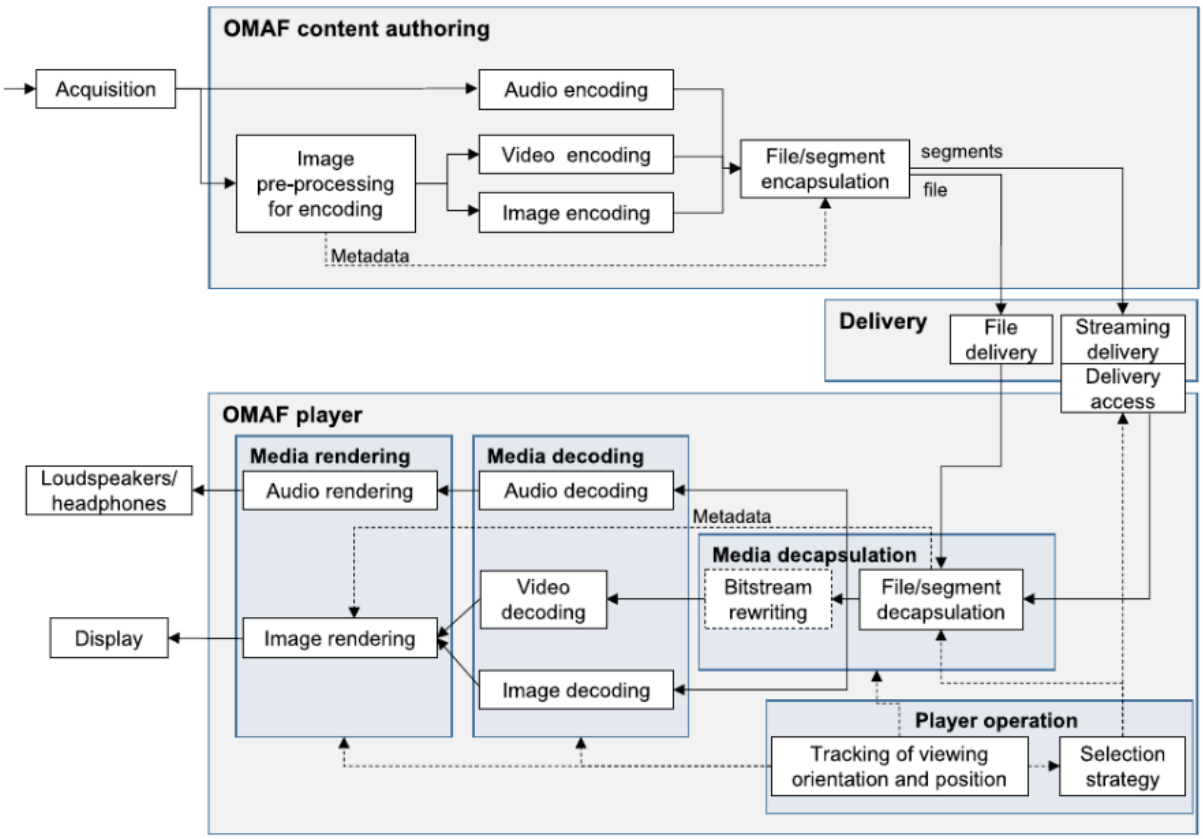

Figure 25: OMAF architecture

The key underlying technologies for file/segment encapsulation and delivery of OMAF are ISO Base Media File Format (ISOBMFF) and Dynamic Adaptive Streaming over HTTP (DASH). OMAF specifies the following:

- A coordinate system that consists of a unit sphere and three coordinate axes; x (back-to-front), y (side-to-side) and z (up).
- A projection and rectangular region-wise packing method used for conversion of a spherical video sequence into a two-dimensional rectangular video sequence. The spherical signal is achieved by stitching video signals captured by multiple cameras.
- Storage of omnidirectional media and the associated metadata using ISOBMFF.
- Encapsulation, signalling, and streaming of omnidirectional media in MPEG-DASH (Dynamic Adaptive Streaming over HTTP) and MMT (MPEG media transport).
- Media profiles and presentation profiles that interoperability and conformity points for media codecs as well as media coding and encapsulation configurations that may be used for compression, streaming and playback of the omnidirectional content.

With relation to the project and more specifically with the requirements of the use-cases, we must look at approaches for omnidirectional video streaming and determine which of the building blocks provided by OMAF are required with relation to the system architecture. There are several publicly available implementations compatible with OMAF v2 [67]. However, other complementary standards, such as Network-based Media Processing (NBMP), outlined below, may also be necessary to fully meet the use-case requirements.

Network-based Media Processing (NBMP)

It is widely agreed that Edge Computing provides more scope for the development of new service platforms. In parallel to this, container-based deployments in the cloud work to simplify things further, especially when combined with software defined networking (SDN). In this regard, easily scalable computing capacity is a requirement for such architectures. Furthermore, hosting services closer to consumers at the network edge should greatly reduce latency and bandwidth requirements for real-time omni-directional media processing.

The Network-based Media Processing (NBMP) standard (ISO/IEC 23090 Part 8)[7] works to facilitate this by defining interfaces, media formats, and metadata to provide a standardised way to perform media processing on any Edge and Cloud computing architectures. This may be useful for the CHARITY Use Cases that perform complex media processing such as content stitching, pre-rendering, point cloud aggregation and timed metadata. NBMP can also be used to reduce network redundancy at delivery by implementing conversion on-the-fly in the network. This may be especially useful for Use Case 3 to enable Mixed Reality (MR) interactive applications that rely on fast world simulation, very low network communication latency and coherency of the simulation across geographically distributed actors.

[7] https://mpeg.chiariglione.org/standards/mpeg-i/network-based-media-processing

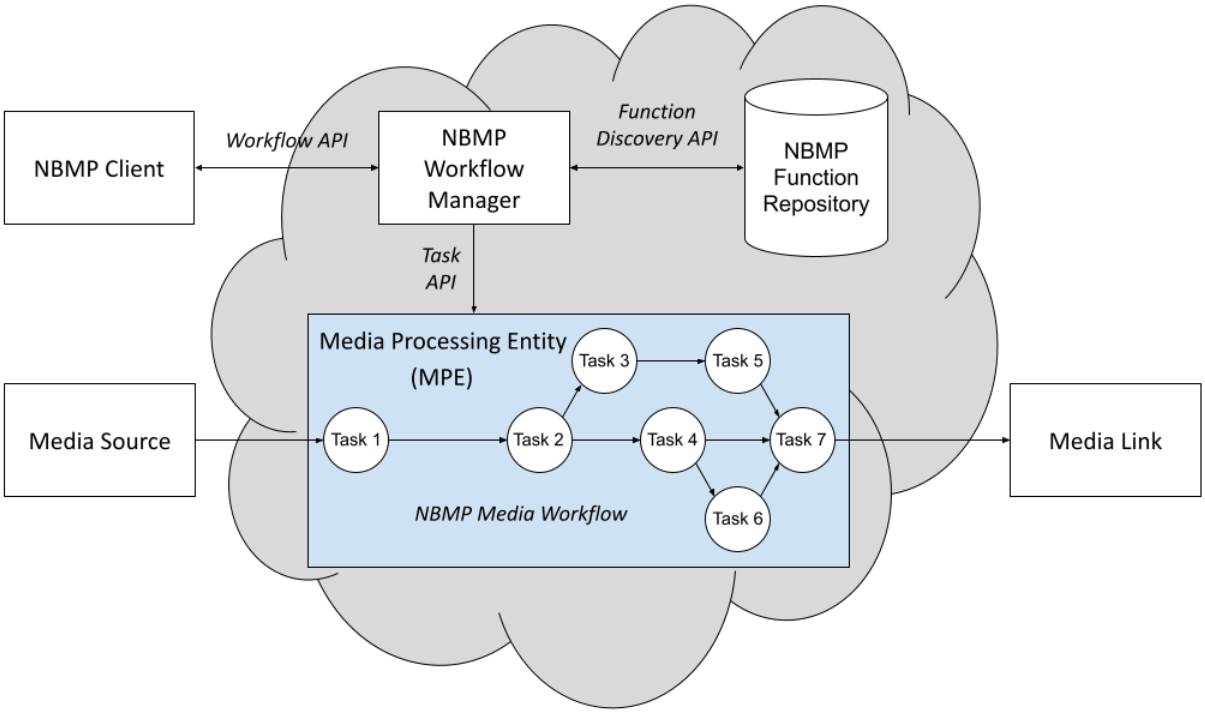

Figure 26: The workflow of Network-based media processing

As shown in Figure 26, users describe the media processing operations to be performed by the media processing entities in a network. A workflow is described by composing a set of media processing functions accessible via the NBMP APIs. The Media Processing Entity (MPE) then runs tasks to process the media data and metadata received from the media source (or other processing tasks), to be consumed by a media sink (or other processing task)[8].

2.9.5 Supporting standards by Khronos group

2.9.5.1 OpenXR

OpenXR[9] is an open standard by Khronos group to connect XR devices and game engines with each other. The goal is to eliminate the various specific implementations to support an XR-device X in game engine Y on platform Z by providing a generic, cross platform application interface to any supported XR device within any supported game engine on multiple platforms (see Figure 27. Special sensors and interfaces for hand tracking and eye-tracking are also supported. Even cloud-based solutions via 5G have already been discussed. Currently version 1.0 has been released. A lot of well-known companies are members of the consortium.

[8] https://www.iso.org/standard/78479.html

[9] https://www.khronos.org/openxr

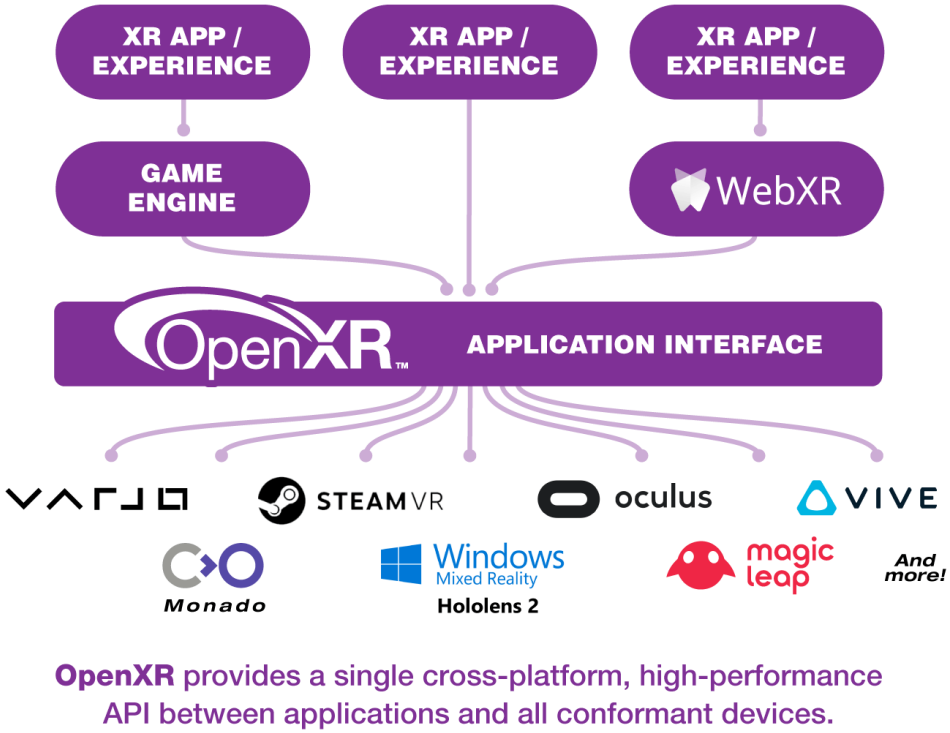

Figure 27: Overview of the Application interface concept[9]

To adapt this standard for a specific end user device like a holographic 3D display, a specific implementation in OpenXR is mandatory to be developed first, so that the various game engines and OpenXR core libraries can access and control the holographic 3D end user device. 3D content in the supported game engines must be appropriately generated to be compatible with holographic 3D. On the other hand, some well-known 2D stereo based HMDs are already supported, so use case owners planning for such types of devices within their CHARITY applications could basically make use of OpenXR to increase the bandwidth of supported end user devices.

2.9.5.2 glTF

glTF™ (GL Transmission Format) is a royalty-free specification for the efficient transmission and loading of 3D scenes and models by engines and applications (Figure 28). glTF minimizes the size of 3D assets, and the runtime processing needed to unpack and use them. glTF defines an extensible, publishing format that streamlines authoring workflows and interactive services by enabling the interoperable use of 3D content across the industry[10].

[10] https://www.khronos.org/gltf/

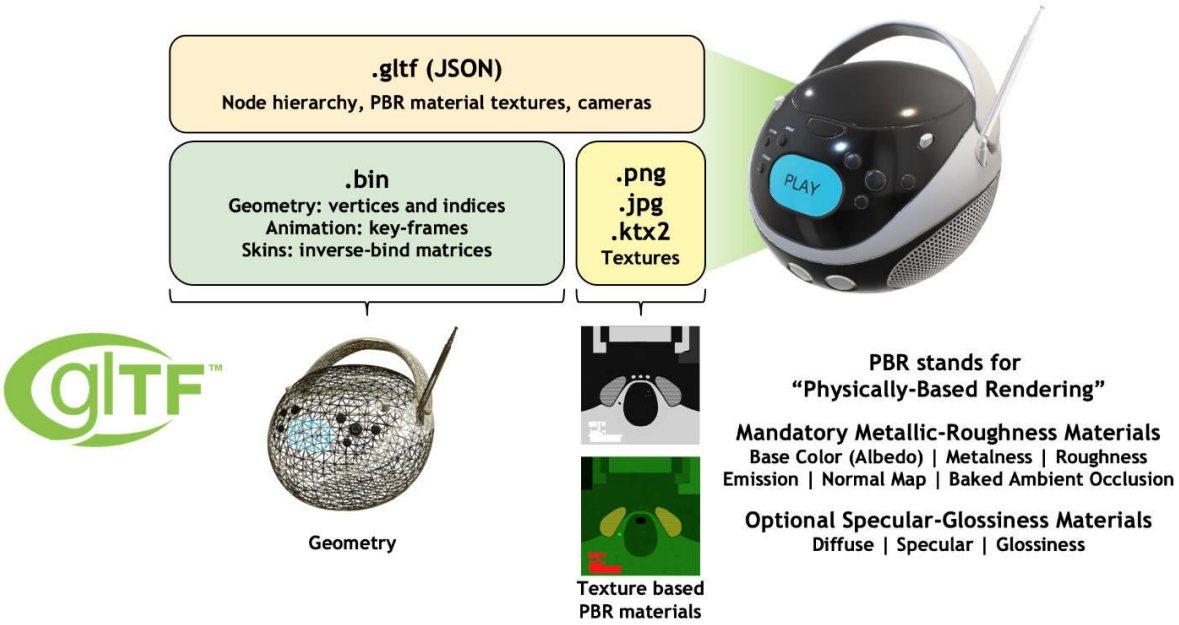

Figure 28: glTF 2.0 Scene Description Structure

2.9.6 DRACO

3D graphics are an indispensable part of many applications, including gaming, design and data visualization. Year by year, graphics processors, rendering engines and creation tools improve, resulting in larger and more complex 3D models. Such high-fidelity models have become the industry standard and help fuel new applications in immersive virtual reality (VR) and augmented reality (AR). However, the increased model complexity forces an increase of storage and bandwidth requirements to keep up with the explosion of 3D data.

To deal with this rising problem, multiple compression algorithms are being designed and implemented. One of the current state-of-the-art industry-ready methods is Draco, an opensource compression library designed by Google's Chrome Media team[11]. Draco aims to improve the storage and transmission of 3D graphics by compressing meshes and point-cloud data.

The huge impact of Draco is depicted in current bibliography, as it influences and drives the design of alternative compression/decompression techniques [68-77]. The framework uses a kd-tree to efficiently store data corresponding to points, connectivity information, texture coordinates, colour information, normals and any other generic attributes associated with geometry. Therefore, it can be used to compress and decompress geometric meshes as well as point clouds, making it ideal for AR and VR applications.

Draco compress *.obj models into *.drc equivalent models and vice versa. Despite the high-compression rate (e.g., 96.7% for the Stanford Dragon and 72.6% for the Stanford Bunny), a model that is compressed and then decompressed preserves the high-fidelity of the original model. The running times of these processes depend on whether C++ or Javascript encoders/decoders are used (e.g., decoding using C++ is significantly faster).

Within the CHARITY platform, a vast amount of AR and VR scenes have to be transmitted, mainly in the form of point clouds. Draco will accelerate such processes by substituting the original models transmitted with the respective compressed ones. In this way, VR and AR scenes will be transmitted and therefore rendered faster, greatly enhancing the quality of experience of the end-users, without compromises on the fidelity of the signal.

[11] https://google.github.io/draco/

2.9.7 ETSI Augmented Reality Framework

ETSI, as part of its standardization efforts, is currently specifying an Augmented Reality Framework (ARF) whose objective is to provide a transparent architecture for interoperation in the highly heterogeneous ecosystem of providers and technologies that stimulate developers.

ARF builds blocks across three layers as depicted in Figure 30: Hardware Layer, Software Layer and Data Layer [78]. The Hardware Layer comprises the Tracking Sensors (e.g., for positioning and orientation), the Processing Units (i.e., dedicated embedded processing components such as GPUs), the Rendering Interfaces (i.e., the components on which the AR content is rendered) and the Interaction Interfaces used to interact with the system. The Software Layer encompasses the Vision Engine, leveraging the output of tracking sensors and processing units to understand the real-world environment, and the 3D Rendering Engine, used to maintain a 3D view from the world. Finally, the Data Layer is divided into a World Knowledge component, which maintains a digital representation of the real world, and the Interactive Contents (i.e., the representation of virtual elements).

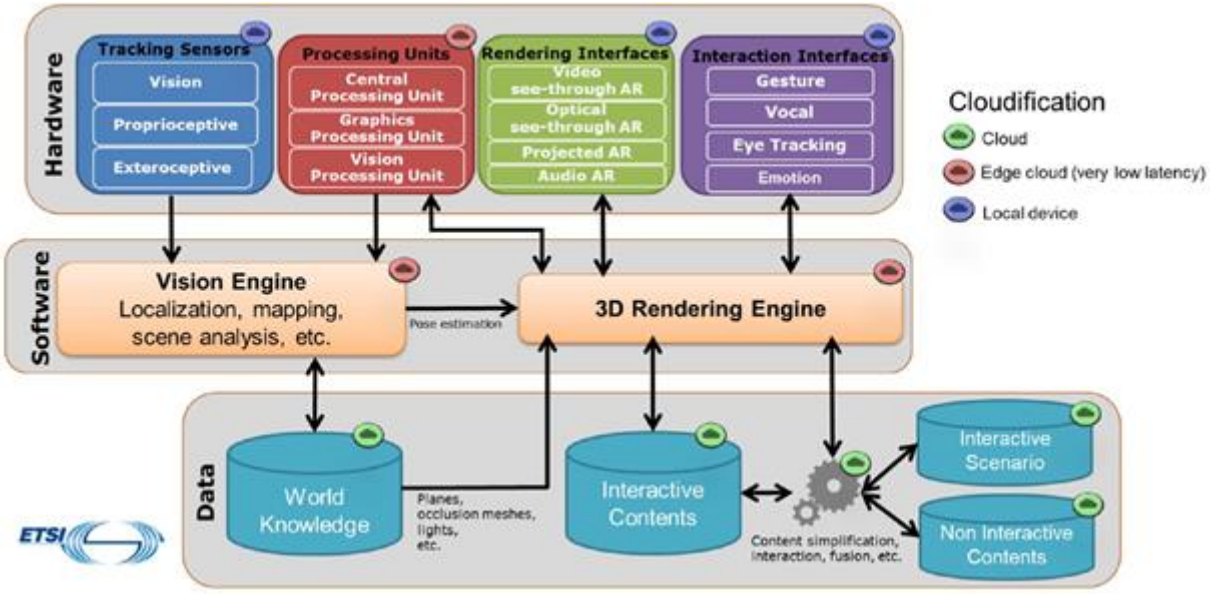

Figure 29: Global overview of the architecture of an AR system [72]

A reference AR functional architecture was also specified in the same document by the ISG AR group. This architecture targets both fully embedded AR systems and implementations spread over IP networks (e.g. edge/cloud environments) [72]. As depicted in Figure 30, the AR functional architecture was organized into ten high-level functions

- World Capture (hardware layer): analysis of the environment using different types of sensors (e.g. RGB-D cameras, event-based cameras) to provide different streaming types (e.g., audio, video, events) to be consumed by additional components and sub-functions. It collects position and orientation and records the space and sounds.
- World Analysis (software layer): processing of the information captured by the sensors. It defines how the AR system shall deliver functions such as Object Recognition and Identification (e.g., using Machine Learning techniques) or Object Tracking capabilities (position and orientation estimation) based on previously captured data. All these parameters allow to create the 3D representation.
- World Storage (data layer): reception and delivering of data used by other functions. It keeps an updated representation of the real world that can be shared between different devices. Also, World Storage can extract the information needed from the representation to update in real time without bandwidth waste. This allows to update the representation faster only using the data that changes when the device is relocating.

- Asset Preparation (data layer): it provides multimedia objects to add to the AR scene and allows to interact with real objects in the scene. Object behaviour is the subfunction that stores predefined behaviours, based on AI algorithms.
- External Application Support (hardware layer): real-time communication and data exchange.
- AR Authoring (data layer): information and format optimization before sending it to the rendering functions. It also adds objects to the scene and their behaviour.
- User Interactions (hardware layer): it delivers to the system information from other type of interaction devices, as tactile surfaces, biometric sensors, etc.
- Scene Management (Software Layer): it is the core function, maintaining scenes at runtime. It receives optimized information from AR Authoring and interactions captured by sensors, and answers with the new interactions that must be shown in the scene.
- 3D Rendering (Software Layer): real-time renderization of the scene with the information provided by Scene Management. It generates audio, video and haptic responses of the interactive objects.
- Rendering Adaptation (Hardware Layer): it updates the scene with the new rendered information.

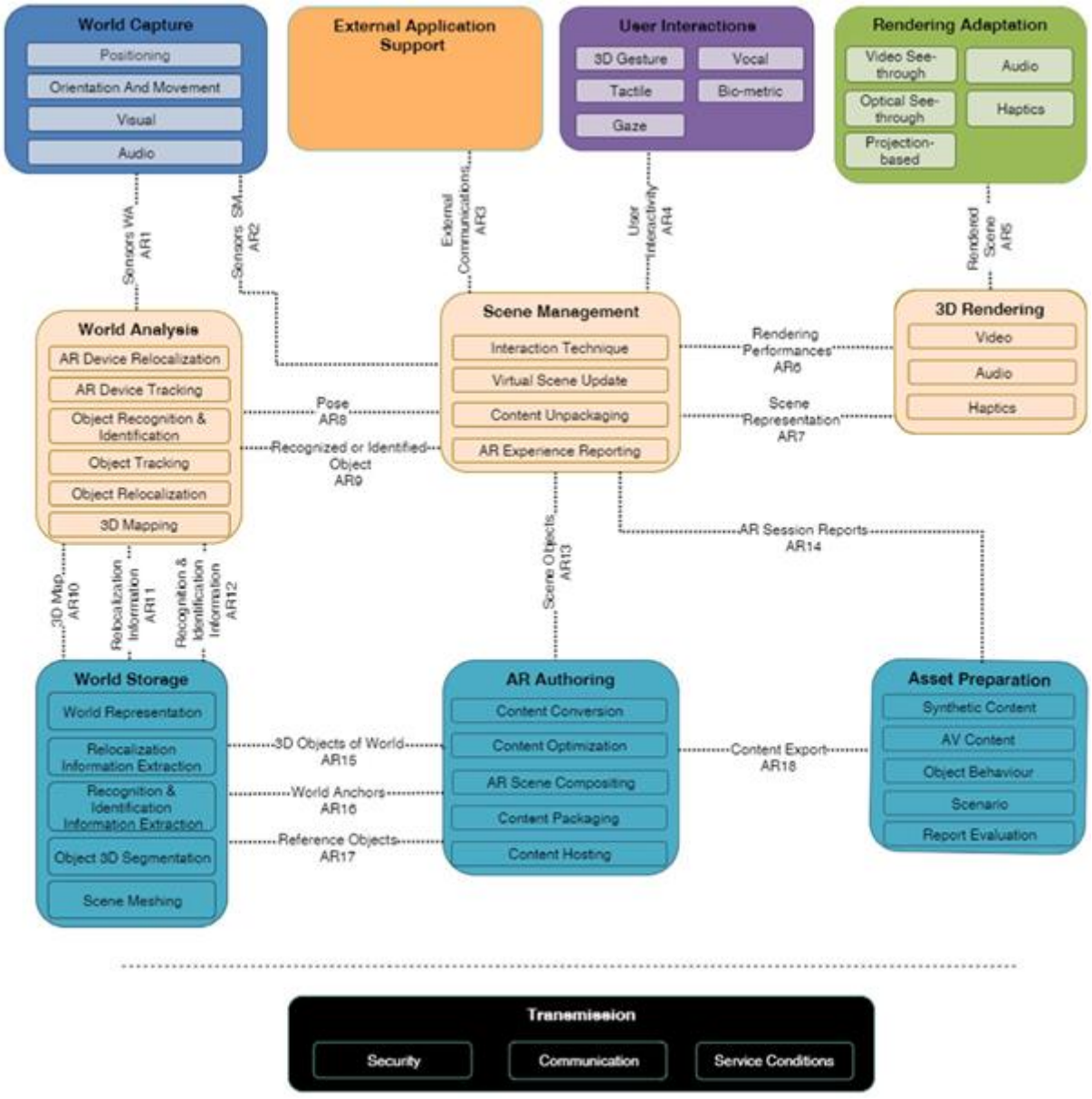

Figure 30: ETSI AR Functional Reference Architecture [72]

Transmission related sub-functions such as Security (i.e., the need for encrypted communications and access control mechanisms), Communication, including the encoding/decoding schemes to cope with large representation formats (e.g., 3D Point Cloud) or even Service Conditions are briefly discussed. Namely, the Service Conditions sub-function, like the CHARITY approach, already comprises the idea that service-related parameters such as network conditions should be taken into consideration to adapt the behaviour of the AR system and be used to optimize the overall Quality of Experience (QoE). Moreover, the presented AR functional framework does also acknowledge the manifold benefits of real-time connectivity with external systems. Nevertheless, the details of such communications are left outside of ISG AR framework specification.

It is also important to note that, such AR system already considers the advantages of a hybrid edge/cloud environment. In the Hardware Layer, most of the components are local. Nevertheless, the dedicated Processing Units were conceptually pushed to a low latency edge cloud. Likewise, the engines present in the Software Layer were also designed as part of the edge cloud. Whereas the remaining components, at the data layer, were designed as part of the cloud.

ETSI specifies relevant components and interfaces required for an AR solution through a set of six use cases centred in industry, whose objective is reducing costs using virtual prototypes and decreasing time-frames of procedures:

- Try before buying with AR, an app to visualize furniture in a certain space;
- Maintenance Support, a tool that improves communication between engineers and technicians while updating circuits in street cabinets;
- Manufacturing procedure, an AR based appliance for workstations with virtual tools to guide user through manufacturing steps;
- Collaborative design review, an app with two headsets that allows to see the same parts of the product in review;
- Factory inspection based on an ARCloud, it detects sensors located in a plant and shows the information to the inspector through a headset;
- Usability Evaluation of Virtual Prototypes, to test early versions before production by observing the virtual interaction that a potential user does.

Each use case contains a description of how their functional steps map into the proposed AR architecture. For some of them, again, it was stressed the benefits of having different functions running on local, edge or cloud, which ultimately allows a better resource exploitation. This is a key aspect which is also explored in CHARITY.

Furthermore, such specification was mainly focused on AR technology and scenarios, yet it could also be interesting to see how such work compares with the Virtual Reality use cases or whether a unified approach for Mixed Reality Scenarios can be devised.

2.9.8 3D Point clouds

2.9.8.1 Overview of 3D Point cloud data formats

Convenient ways to provide visual information are based on image- or video data. This is due to the fact that 2D displays or 2D projection systems are a common way to present visual information. They use pixelated image-data as input, with a certain update rate we get video information. Various methods exist to compress image or video data. The basis for image compression is on the one side the fact that the human eye is somewhat tolerant against some variation in information. On the other side there is often a lot of redundant information in images and video material. So this type of data can be compressed very efficiently.

But with upcoming new display technology in the area of volumetric or holographic displays, the demand for new data formats beyond only flat 2D surfaces comes up. Also, the demand for variable perspectives within content arises - e.g. for sports events recordings. A format providing not only pixels

for representation on a 2D surface but providing a volumetric representation based on 3D points to allow various perspectives would be an excellent basis for such scenarios - a 3D point cloud. Another known term for such 3D points is called Voxels.

There are various approaches to implement such a format. From the view of official standardization, good progress was made by the MPEG Point Cloud Compression project [79]. It was initiated in 2014. A call for proposals in 2017 resulted in a first draft of the standard by the end of 2018. Until today, the standard is under development, there is an actively maintained reference implementation. Basically, the standard proposes two types of 3D point cloud compression - video based (V-PCC ISO/IEC 23090-5) and geometry based (G-PCC ISO/IEC 23090-9).

Figure 31: Example data sets used for comparing V-PCC[12]

The V-PCC variant uses classic image-based processing (colour + depth + occupancy maps). By applying common image-based compression methods (HEVC in the reference implementation), quite good compression rates have been achieved. The method is based on projection of the 3D source scene or point cloud on multiple 2D maps from different perspectives. These projections or patches are then mapped into the frame - the "atlas" - to be encoded / decoded by means of video compression. Here multiple maps are generated, attribute maps (can be RGB colour but also something else), depth maps (representing the distance from the according perspective) and an occupancy map (representing valid pixels). Within a (lossless encoded) meta data channel, information about how to reconstruct these patches back into the 3D point cloud are provided within the multiplexed data stream. Within the process of generating the patches and atlas, some improvements on the data are done, e.g. detection and removal of duplicate 3D points or improvement of quality especially on the regions between patches (seams). As a result, very good compression rates have been achieved. The MPEG PCC research group defined some reference data sets (see Figure 31Figure 31), where the rates and quality of different algorithm versions and parameter variants could be measured and compared. For example, a scene with 100k points @30fps corresponds to 360 Mbit/s uncompressed data rate. With V-PCC a compression to about 1 MBit/s can be achieved using version TMC2v8.0 while achieving good quality.

The G-PCC variant is based on compressing the 3D points directly one by one. Here the 3D points structure (point locations) is encoded lossless by using an octree approach. Here a cube is dived into 8 cubes, iteratively, from top to bottom, until finally the point level is reached. At each level it is noted if there are some valid points inside the cube (appropriate bit is 1) or not (bit is 0). For one cube, an 8 bit word represents the 8 cubes assigned in the next level of the hierarchy. In contrast, for encoding point attributes (i.e. RGB colour), three compression methods have been developed. These methods basically make use of similarities / redundancy between colours down the octree graph. The algorithm also allows for different levels of details - usable e.g. to adapt for variations in available data rate or to

[12] http://plenodb.jpeg.org/pc/8ilabs/

adapt for current detail requirement in rendering process. Currently, the algorithm does not use temporal compression approaches to enable lower data rates in situations where the 3D scene does not change much from frame to frame - compared to video compression where this approach is extremely effective. But some work into this direction was supposed for the next version of the standard. For an example scene with 100k points at 10 fps corresponding to 110 MBit/s uncompressed data rate, a compression down to about 24 MBit/s could be achieved with good quality.

For CHARITY, especially for the use case “Holographic Assistant”, suitable point cloud formats with the support of real time capable generation, compression and decompression are a strong requirement. This type of content representation is the ideal input for generating high quality 3D holograms to be presented on holographic 3D display devices based on diffraction and interference of light. Currently, the existing standards are not 100% compatible with CHARITY requirements. It is needed to check whether there could be some extensions added or a simplified solution developed and optimized for operation in the CHARITY cloud. Real time capability / high performance and compatibility with holographic 3D are some of the key requirements.

2.9.8.2 3D Mesh generation from 3D Point cloud data

Mesh generation (Figure 32) based on the 3D point cloud data is a very complex process requiring agile algorithms and quite a lot of computing power. Here we focus on approaches dealing with the problem of fast and robust reconstruction of shapes and surfaces rather than very high precision and quality mesh generation.

In the literature, there exists a number of methods for effective 3D point cloud meshing. Some methods utilize a feature detection process and first extracts from the point cloud a set of sharp features. Then the reconstruction process must be incorporated in order to provide implicit surfaces and generate a mesh approximating the surfaces and extracted edges. Such a mesh provides a trade-off between accuracy and mesh complexity but also a trade-off between speed and accuracy. The whole process should be as robust as possible to noise contained in the 3D point cloud.

Alpha shapes is another interesting, well documented and widely used method for mesh regeneration. It was introduced by H. Edelsbrunner et al. in 1983. As a generalization of a convex hull it is quite an intuitive and easy to understand method. It is based on the gradual removal of those parts of the space surrounding the point cloud that do not contain any points. The method can be used iteratively to increase the accuracy of the resulting mesh. More detailed description of the method can be found in[13]. *Ball pivoting* by F. Bernardini et al. and *Poisson surface reconstruction* by M. Kazhdan et al. are other meshing algorithms that need to be mentioned here. They provide different characteristics to the resulting mesh.

There exists an open-source library called Open3D that supports rapid development of software that deals with 3D data - among others it supports mesh generation from 3D point cloud data. More about this library can be found in[14]. Regardless of which mesh generation method we use, the final result depends to a large extent on the quality of the point cloud that we provide as input.

Unfortunately, the very process of creating a point cloud from the real space around us is extremely complex and is potentially burdened with many imperfections. Obtaining data from the image generated by RGB cameras is usually insufficient for rapid meshing. A much better and more precise method is to use dedicated devices and technologies such as LiDAR. Apple built-in LiDAR technology into their newest mobile devices. They use their own proprietary libraries/APIs to perform fast and robust mesh generation. The documentation of the methods and algorithms they use is unfortunately not publicly available. Nevertheless, it is currently the fastest and most precise technology available

[13] https://graphics.stanford.edu/courses/cs268-11-spring/handouts/AlphaShapes/as_fisher.pdf

[14] http://www.open3d.org/docs/latest/index.html

for end-users. It provides a very effective solution to both problems: generating 3D point cloud data from the real surroundings and mesh generation.

Figure 32: An example of a scanned environment

The LiDAR Scanner measures the distance to surrounding objects up to 5 meters away, works both indoors and outdoors, and operates at the photon level at nano-second speeds. New depth frameworks in iPadOS combine depth points measured by the LiDAR Scanner, data from both cameras and motion sensors, and is enhanced by Apple proprietary computer vision algorithms for a more detailed understanding of a scene.

2.10 Other EU Project Architectures

In this section, we introduce a number of architectures that were devised in EU projects, and that played a fundamental role in inspiring the design of the CHARITY architecture, as introduced in Section 3.

2.10.1 COLA - Cloud Orchestration at the Level of Application (MiCADO/Occopus)

MiCADO*scale* is a multi-functional, cloud-agnostic orchestration and auto-scaling framework for Kubernetes deployments and is a primary result of the H2020 EU funded project COLA. MiCADO*scale* supports auto-scaling at two levels (Virtual Machine and Container). For VM, a built-in Kubernetes cluster is dynamically increased or decreased by adding or removing virtual machines in the cloud. For Kubernetes, the number of replica pods tied to a specific Kubernetes deployment can be increased or decreased. In short, MiCADO*scale* enables applications to automatically deploy, scale, manage and monitor containerized microservices orchestrated by Kubernetes. In Figure 33, MiCADO*scale* high level architecture is shown.

A TOSCA-based Application Description Template (ADT) is created and used to describe an application in MiCADO. These templates are used to achieve portability across different cloud environments, minimising the possibility of vendor lock-in. MiCADO*scale* supports a variety of cloud middleware,

including commercial cloud providers such as Microsoft Azure, AWS and CloudSigma, as well as open source cloud software such as Openstack and OpenNebula.

The MiCADO core services are deployed on a virtual machine (the MiCADO Master node), via an Ansible playbook. The MiCADO Master runs the Kubernetes control-plane, the Docker runtime, Terraform and/or Occopus (to provision and scale VMs), Prometheus (for monitoring), the MiCADO Policy Keeper (for automated scaling) and the MiCADO Submitter (the submission endpoint). Where possible, these components are themselves deployed to Kubernetes in a microservices architecture.
At run-time, MiCADO workers (realised on new VMs) are provisioned and instantiated on-demand and run the Prometheus Node Exporter and Google cAdvisor to collect a default set of metrics used for scaling the infrastructure. The newly instantiated MiCADO workers join the Kubernetes cluster managed by the MiCADO Master and application containers are appropriately scheduled across those workers.

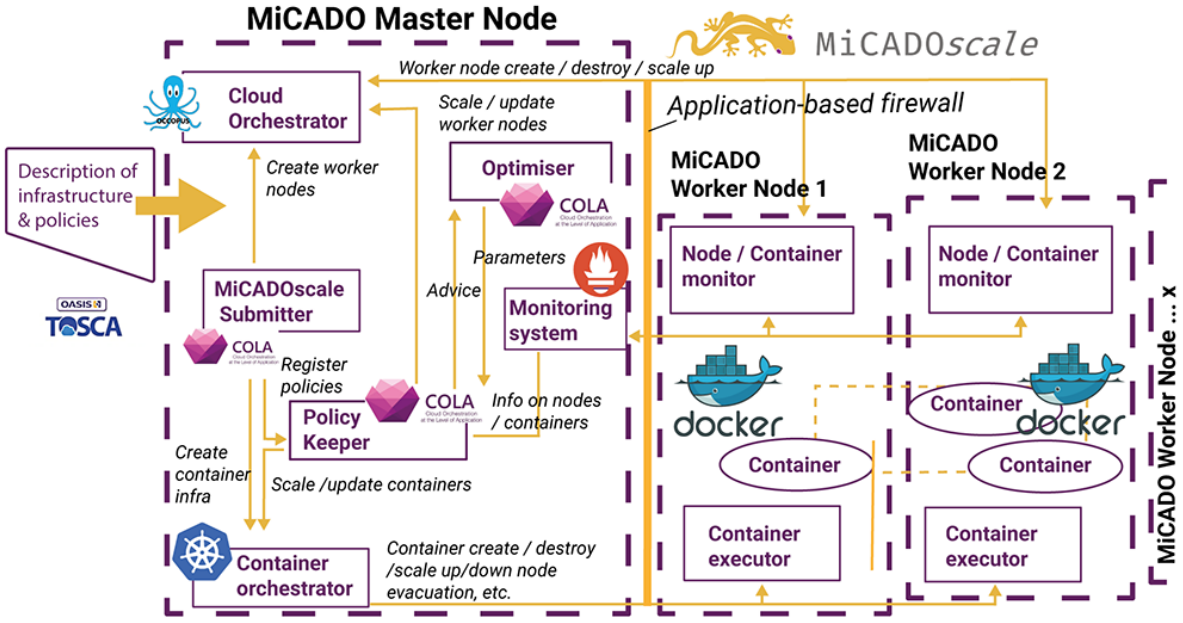

Figure 33: MiCADOscale high-level architecture[15]

Since the project's completion, MiCADOscale development has been carried forward by the original developers, University of Westminster and SZTAKI (Institute for Computer Science and Control), and the product has been commercialised. Support is provided by CloudSME a commercial entity originating from another H2020 EU funded project; CloudSME - Simulation for Manufacturing and Engineering.

2.10.2 ANASTACIA - Advanced Networked Agents for Security and Trust Assessment in CPS / IOT Architectures

2.10.2.1 Overview

The ANASTACIA project aimed to develop a holistic security framework for IoT infrastructures. Leveraging new monitoring methodologies and tools, it is able to take autonomous decisions to provide dynamic security. The ANASTACIA platform is built upon SDN controllers, NFV orchestration platforms and IoT controllers. Using these three technologies, ANASTACIA offers an IoT infrastructure where the streams from/to IoT devices can be dynamically monitored, processed and routed in order to ensure a secure platform.

[15] https://micado-scale.eu

Figure 34 depicts the architecture of the ANASTACIA framework. It is composed of five planes that enable the platform to function in an autonomic and intelligent way. This allows the ANASTACIA platform to offer a dynamic behaviour by enforcing dynamic security policies and mitigation strategies when facing threats.

The Security Orchestrator Plane is responsible for translating high level security policies into concrete orchestration actions that should be applied on the infrastructure. It manages, orchestrate and configures the security appliances. These security appliances can be VNFs or PNFs; they can consist in network functions (e.g. routing), security functions (e.g. firewall, VPN) or monitoring functions (e.g. IDS, antivirus). In short, the Security Orchestrator Plane has the whole vision of the underlying infrastructure and the functions running therein.

The main responsibility of the Security Enforcement Plane is to connect the ANASTACIA platform with the IoT platform. It is the interface with which the Security Orchestrator Plane can orchestrate and manage the underlying security functions by enforcing the configurations and reactions initiated by the Security Orchestrator Plane. Indeed, it is up to the Security Enforcement Plane to instantiate and perform the lifecycle management of the security functions running in the platform.

The Monitoring and Reaction Plane collects and processes the monitoring data that is produced by the IoT infrastructure and the security functions running in the platform. This plane offers an intelligent, data-driven and automated monitoring of the whole infrastructure. Leveraging intelligent monitoring algorithms, this plane can detect threats and anomalies, raise alerts and also setup mitigation strategies, in terms of reconfigurations and alerts sent to the Security Orchestration Plane and/or to system administrators.

Finally, the User Plane and the Seal Management Plane interact with the system administrator. The former offers tools and applications to help him manage the security of the IoT platform. The latter provides him with a real-time indicator of the security level of the architecture.

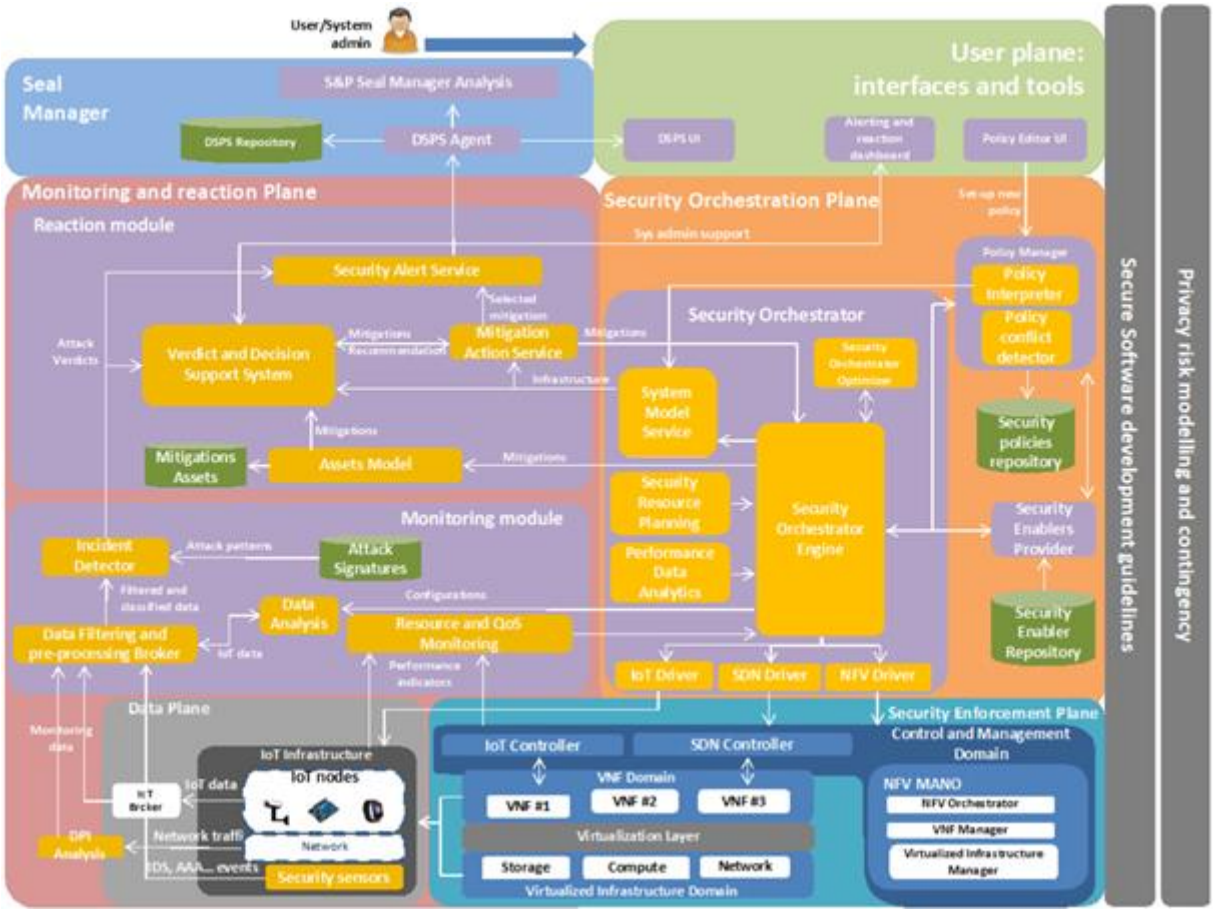

Figure 34: ANASTACIA architecture design [80]

2.10.2.2 Discussion

The ANASTACIA platform is targeted toward the security of IoT platforms. It is intended to be installed above an IoT platform. The CHARITY architecture is heavily inspired from the planes of ANASTACIA architecture. Except that the main difference is that CHARITY architecture is not only concerned by the security concerns of XR services but it is meant to the whole management of XR services including security aspects. Another point is that CHARITY is meant to be deployed over multiple administrative and technological domains while ANASTACIA is concerned with one domain only.

2.10.3 INSPIRE-5Gplus - INtelligent Security and PervasIve tRust for 5G and Beyond

INSPIRE-5Gplus aims to design a zero-touch end-to-end smart network and service security management framework. This framework provides protection and ensures trustworthiness and liability when managing 5G network infrastructure across multiple domains. INSPIRE-5Gplus closely follows the key principles of ETSI ZSM reference architecture. It is split into several security management domains (SMDs) which results into a robust architecture and also ensures separation of concerns (See Figure 35). The SMDs are composed of multiple security services which constitutes a service-based architecture where each SMD is responsible for the security of resources and services within its scope. As it can be seen in Figure 35Figure 34, an SMD consists in a Security Data Collector, a Security Analytics Engine, a Decision Engine, a Security orchestration, Policy and SSLA Management, and Trust Management. All these security services are exposed and can be consumed through the integration fabrics which by design ensure authentication, authorization and access control. An E2E SMD is used to ensure the management of security functions for E2E services that span multiple domains. All of these SMDs operate in an intelligent closed-loop fashion which enables AI-driven software defined security (SD-SEC) orchestration and management in compliance with the expected Security Service Level Agreement (SSLA) and regulatory requirements.

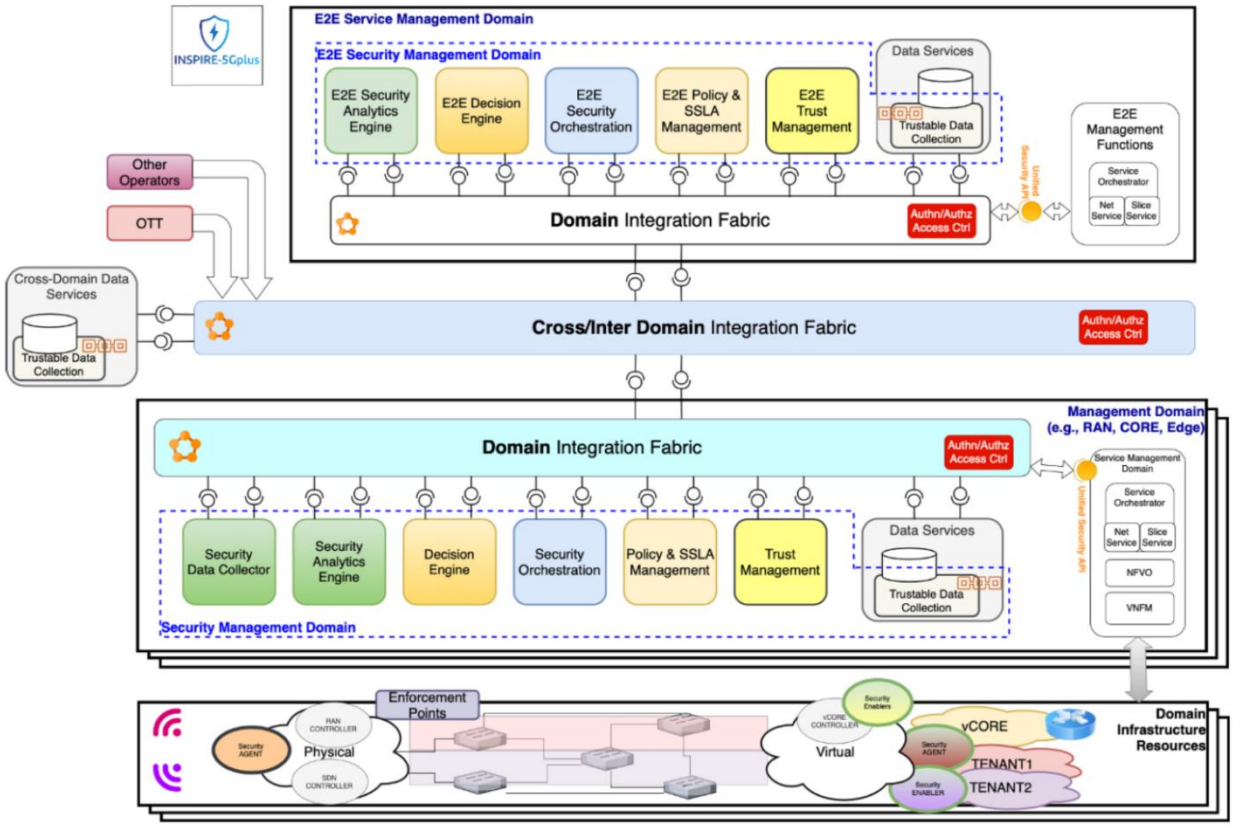

Figure 35: INSPIRE-5Gplus High-Level Architecture [81]

Figure 36 shows the intelligent closed-loop in which the SMDs operate. It can be regarded as the integration of AI/ML techniques into a combination of the stages of OODA (Observe-Orient-Decide-Act) and MAPE-K (Monitor-Analyse-Plan-Execute Knowledge) models.

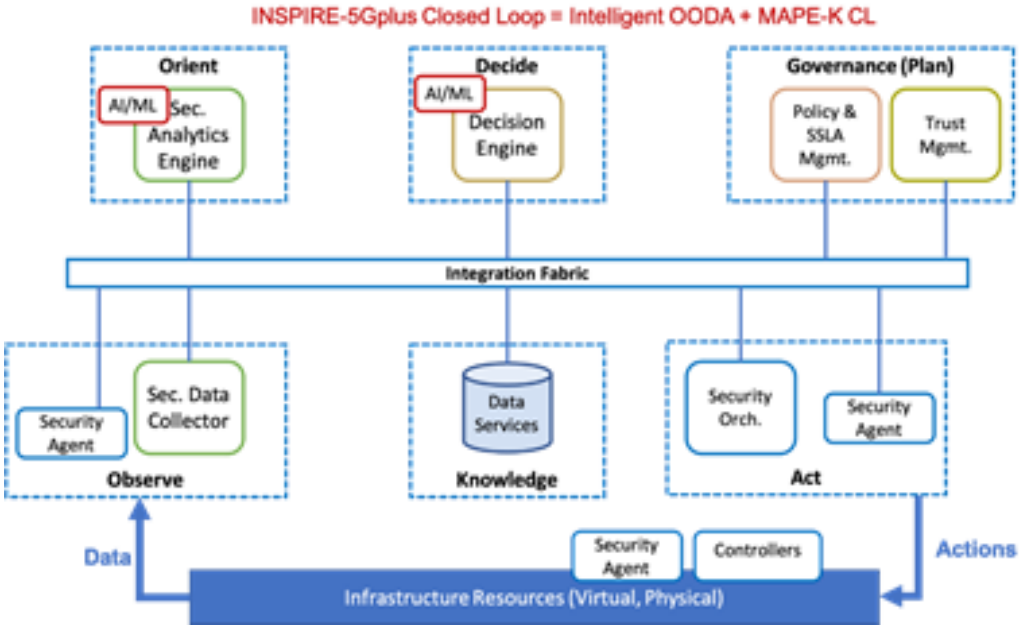

Figure 36: INSPIRE-5Gplus Framework Closed Loop Model [81]

2.10.4 MonB5G - Distributed management of Network Slices in beyond 5G

2.10.4.1 Overview

As described in detail in [82], MonB5G aims to design a novel framework for accommodating the provisioning, deployment, and lifecycle management (LCM) of large numbers of network slices, in alignment with the vision of 5G and beyond. For achieving a high level of efficiency and scalability, it adopts the MAPE (Monitor-Analyse-Plan-Execute) paradigm and relies on distributed closed feedback loops, facilitated by AI-driven operations, and that is to ensure a certain degree of autonomic network operation. Such an approach facilitates local data processing, and consequently allows rapid data-driven decisions which are crucial for handling time-critical issues. Network components become much more scalable and agile, since they are now operating without the overhead of transferring large datasets of information, thus reducing the delay, and optimizing the usage of valuable system resources.

The AI-based MAPE management loops are implemented using four components, namely: The Monitoring System (MS), Analytics Engine (AE), Decision Engine (DE) and the actuation functions (ACT). The AEs and DEs leverage modern distributed AI (DAI) techniques, including federated learning and multi-agent deep reinforcement learning, to empower autonomous distributed slice LCM capabilities. The DAI techniques allow distributed learning and decision making while sharing the learned knowledge between agents, hence significantly reducing the learning overheads while increasing the accuracy.

To deliver the highest value possible while maintaining an inherent system simplicity, the MonB5G framework is designed based on specific principles. Particularly, the architecture: (i) maintains a strong distinction of components, having domain-grade slice LCM and resource management, both handled by entities which remain agnostic to the slices per se, while each slice integrates its own management platform as dictated by the In-Slice Management (ISM) paradigm, (ii) delivers hierarchical, end-to-end slice orchestration capabilities together with scalable and programmable slice management, by fully integrating ISM implemented as a set of VNFs which are then responsible for the fault, configuration, accounting, performance, and security management (FCAPS) of the specific slice, (iii) supports distributed, AI-driven management operations, (iv) incorporates programmable, energy-aware

infrastructure management, (v) boosts slice security through the slice management isolation delivered by ISM, and (vi) allows the creation of a dedicated "management slice" which can be used for run-time management of multiple slice instances, in alignment with the Management as a Service (MaaS) paradigm. An overview of static components and business actors of the MonB5G architecture is presented in Figure 37.

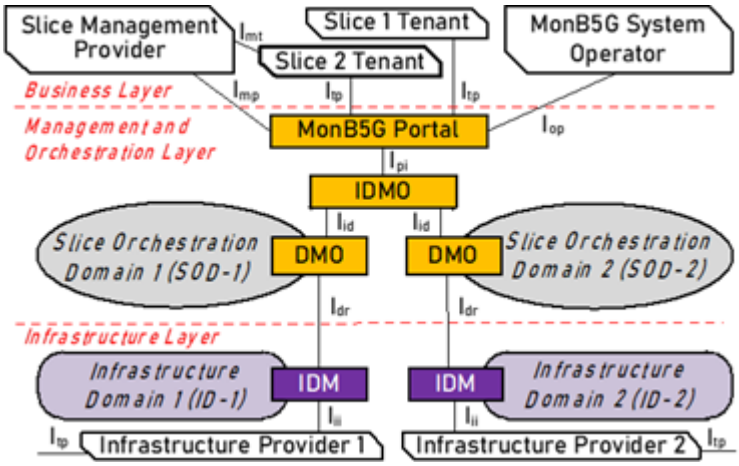

Figure 37: MonB5G Architecture - Static Components and Business Actors [82]

2.10.4.2 Discussion

Lessons learned from MonB5G architecture can be integrated into CHARITY architecture. One of such lessons is the fact that CHARITY seeks to have independent domains with per-domain services orchestration functions and e2e services orchestration entities. CHARITY will use some of the AI and DAI systems offered by MonB5G, and it will also devise new systems that are targeted towards XR services. MonB5G is focusing on the management of a massive number of slices (XR can be considered as one functionality). While CHARITY focuses on the deployment of XR services which have very stringent requirements in terms of computing and network latency and bandwidth. Thus, such services may need special components to ensure the satisfaction of such requirements.

2.10.5 ACCORDION - Adaptive edge/cloud compute and network continuum over a heterogeneous sparse edge infrastructure to support nextgen applications

2.10.5.1 Overview

ACCORDION establishes an opportunistic approach in bringing together edge resource/infrastructures (public clouds, on-premise infrastructures, telco resources, even end-devices) in pools defined in terms of latency, that can support Next Generation application requirements. To mitigate the expectation that these pools will be “sparse”, providing low availability guarantees, ACCORDION will intelligently orchestrate the compute & network continuum formed between edge and public clouds, using the latter as a capacitor. Deployment decisions will be taken also based on privacy, security, cost, time and resource type criteria. The slow adoption rate of novel technological concepts from the EU SMEs will be tackled though an application framework, that will leverage DevOps and SecOps to facilitate the transition to the ACCORDION system. With a strong emphasis on European edge computing efforts (MEC, OSM) and 3 highly anticipated NextGen applications on collaborative VR, multiplayer mobile- and cloud-gaming, brought by the involved end users, ACCORDION is expecting to radically impact the application development and deployment landscape, also directing part of the related revenue from non-EU vendors to EU-local infrastructure and application providers.

ACCORDION assumes each edge or cloud node in a computing continuum as an execution environment for its application components. ACCORDION dubs this heterogeneous resource pool as a “minicloud”, where the lowest level functional component is a VIM (Virtual Infrastructure Manager), abstracting the underlying resources from the higher layers. Miniclouds are then associated into ACCORDION-style

federations. The overarching management layer oversees a federation (continuum management framework,) and provides tools for the deployment as well as development (application management framework) of ACCORDION applications. Figure 38: ACCORDION macro architecture depicts the high-level organization of the ACCORDION architecture.

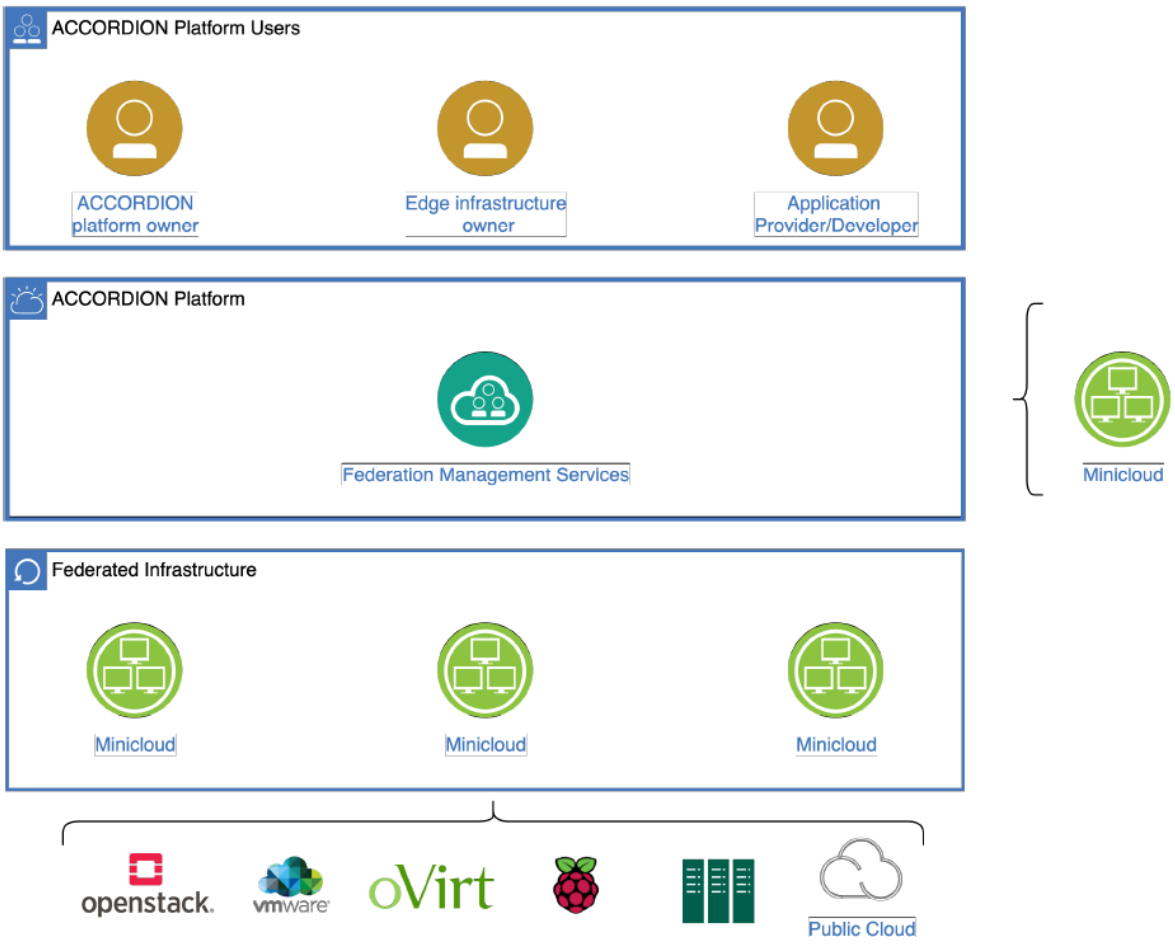

Figure 38: ACCORDION macro architecture

2.10.5.2 Discussion

There are wide chances of taking inspiration and reusing concepts and ideas from ACCORDION. The two projects (CHARITY and ACCORDION) have more touchpoints, starting from the foundational idea of optimal orchestration of virtualised resources in a heterogeneous cloud-to-edge space. The target of the two projects is similar, since XR applications can be considered a special class of NextGen applications. The potential touchpoints include, among the others, application modelling, lifecycle management, computing and network orchestration, implementation of VIM, resource registration and discovery mechanisms. Moreover, the two projects have a number of common partners, which can further facilitate their interaction and reciprocal fertilization.

3 CHARITY General Architecture

In this section, the general architecture of the CHARITY platform is provided. This architecture is built upon some key enablers to support future XR applications. This section will demonstrate the feasibility of the proposed architecture by describing the processes driving the platform with relevant use case scenarios, and mapping the envisioned functionality to existing tools.

3.1 Architectural requirements

The main challenges that the desired architecture is meant to address are the need for ultra-low latency/ultra-high bandwidth and user mobility. However, in order to preserve a complete view of a modern architecture, we extend our research towards the orchestration of network resources, especially in multi-domain orchestration scenarios.

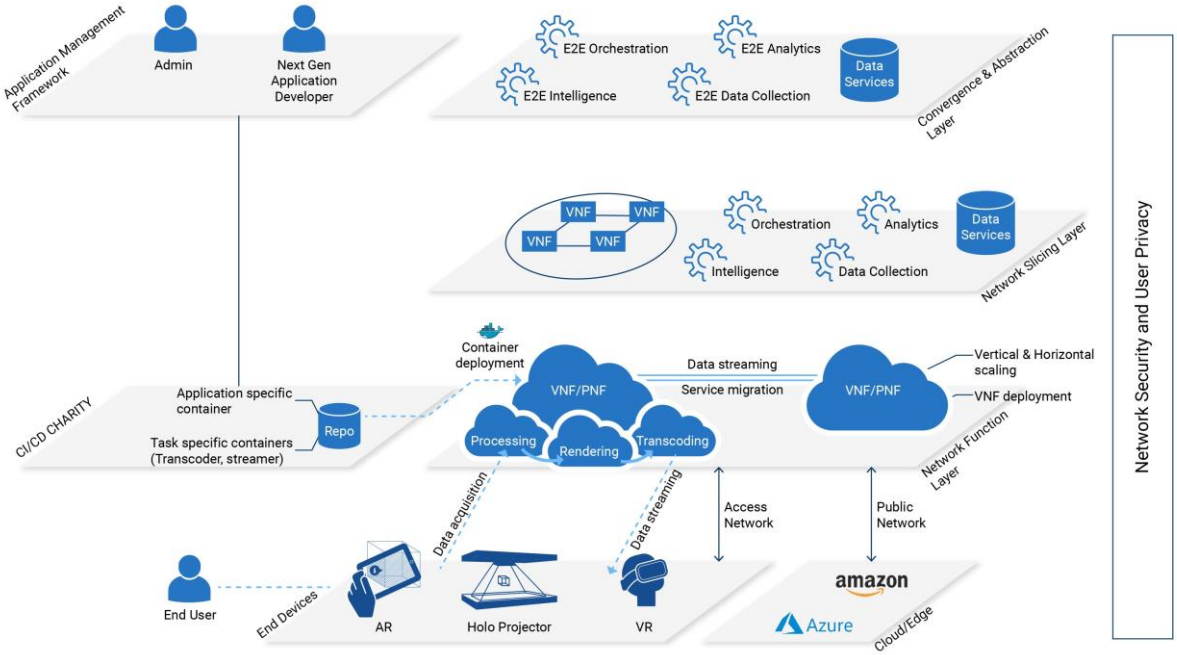

Figure 39: CHARITY's general architecture diagram - as submitted in the description of work

Figure 39 shows the generic architecture diagram of the CHARITY platform. It is composed of five main layers; Infrastructure layer, Network Function layer and CI/CD framework, Network Slicing layer, Convergence and abstraction layer along with the Application Management framework, and finally the Network Security and User Privacy layer.

Infrastructure Layer – consists of all the underlying physical infrastructure that spans from computing platforms to end user devices. These computing platforms can belong to different providers, they can be placed in central clouds as they can be distributed over multiple edges, or even extreme edges. All of this heterogeneity requires an abstraction that would permit the upper layer to seamlessly request and use resources from all the connected computing platforms.

Network Function Layer - is in charge of managing the computations and resources of network functions. This layer would consist in a cloud native platform, where the VNFs are running as a microservice on top of a continuum of edges and clouds. These running service function chains (i.e., stitched VNFs) are smoothly adapted and tailored for the XR services need and also according to available resources. In this layer, the VNFs are monitored and smoothly scaled horizontally and vertically according to the exerted load.

Network Slicing Layer - orchestrates and manages the infrastructure and network functions provided by the two previous layers. Therefore, this layer interconnects the network functions, the edges, and

the clouds and thus forms a slice that spans multiple domains while abstracting and hiding these details to the running VNFs. This layer also contains the automation loop that ensures the creation of a self-orchestrated, self-managed and self-optimized slice that continuously satisfy the desired KPIs and requirements of each XR service. These automation loops, present at each network slice level, consist in data collection analytics, intelligence, and orchestration.

Convergence and Abstraction Layer - offers a platform that allows XP providers to formulate their requests for slices creations via blueprints that specify service architecture along with targeted KPIs and end users' geographic locations. It is an End-to-End (E2E) service management domain that is able to orchestrate and manage all XR services. Indeed, it helps ensure cohesion and unified management and interaction between the different XR services.

Network Security and User Privacy Layer - aims to secure the applications development, the communications, and the platform where Service are running. With this layer and the CHARITY's CI/CD pipeline, a DevSecOps mindset can be pursued, which allows automation and incorporation of security testing and auditing approaches earlier in the software development lifecycle. Similarly, infrastructure hardening should also be supported especially through the use of service mesh. Also, dynamic security can also be provided through the Security as a Service (SECaaS) concept where security VNFs are dynamically deployed to enhance the data privacy, security monitoring of the network communications, and a secure execution over the cloud/edge continuum.

3.2 Architecture outline

Figure 40 depicts the general overview of the architecture as agreed upon among the consortium members. Building up on the expertise and knowledge gained and exchanged among the partners, as surveyed in Section 2, the CHARITY platform follows both the NFV and ZSM frameworks in order to create self-managed E2E network slices. It is composed of three planes: i) the deployment plane, ii) the domain-specific monitoring and reaction plane, and iii) the E2E conducting plane. The deployment plane consists of the infrastructure where the XR services are actually running. It thus hosts the different Virtual Network Functions (VNFs) that are composing the different XR services. The main responsibility of this plane is to manage the computational, network, and storage resources of the infrastructure. The domain-specific monitoring and reaction plane is responsible for monitoring the service inside a technological or administrative domain. This domain level monitoring helps in detecting issues and mitigating them without having to resort to the E2E conducting plane. These local detection and mitigation mechanisms would accordingly lessen the burden exerted on the E2E conducting plane. The E2E conducting plane is responsible for creating the different sub-slices inside each domain and for monitoring the E2E KPIs of the XR services. It is also responsible for shifting the services between the different domains when necessary.

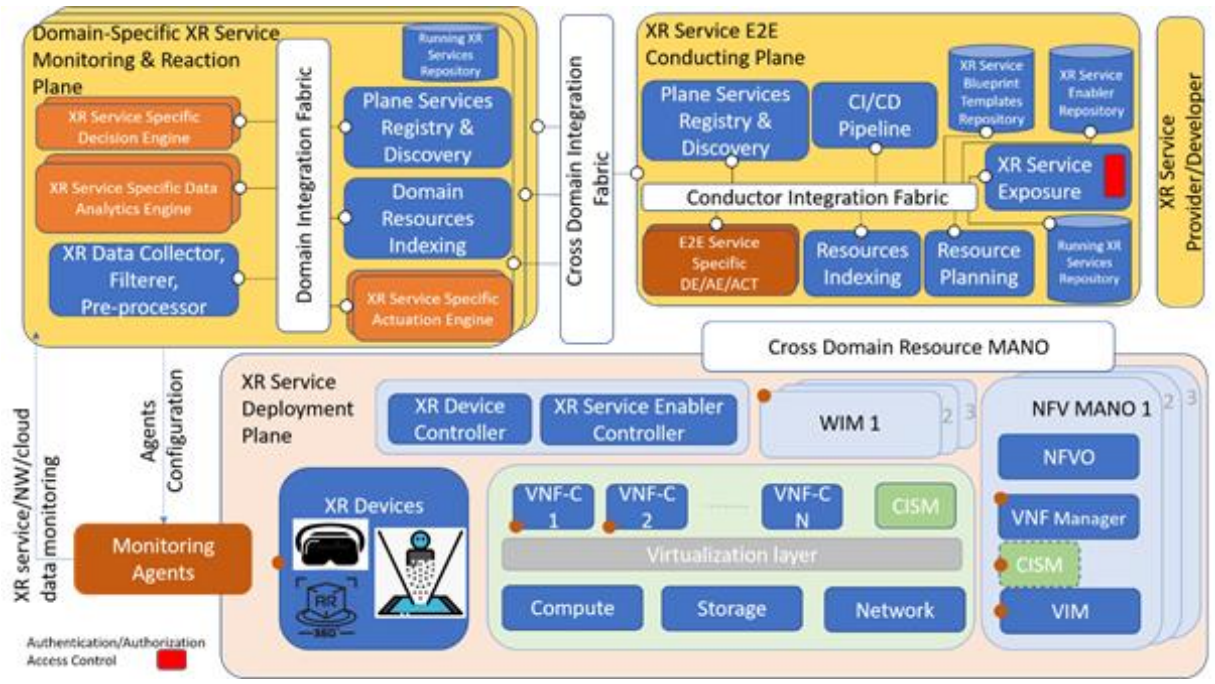

Figure 40: High level architecture of CHARITY platform

3.3 Components of the architecture

In the following sections each plane of the architecture is detailed.

3.3.1 Integration Fabric

The integration fabric enables the inter-operation and communication between the different functions. Through the integration fabric, the different functions play both roles of service consumer and service producer. It allows registration and discovery of services, which means that services should be able to register and be added into a catalogue. Services can discover each other by searching the catalogue for specific capabilities. It also allows the invocation of services, either by a direct request to a specific service or by a request to a class of services (service mesh concept). The integration fabric also offers dedicated communication channels between the different services. All of these features are protected by an authentication and authorization service.

The integration fabrics inside the domain-specific monitoring and reaction plane and the E2E conducting plane help the components inter-plane communication. It allows having default secure communication channels between the different components. The cross-domain integration fabric helps the communications between the domains and the communication with the E2E conducting plane.

3.3.2 XR Service Deployment Plane

The deployment plane hosts the XR services. It is composed of different technological and administrative domains. Indeed, such domains can belong to different entities which may result, for instance, in different charging schemes and also different management APIs. These domains can be also different in the nature of the underlying technology, such as the RAN, edge and cloud domains. Furthermore, it shall be noted that an XR service can have different components (VNFs) running on different domains.

3.3.2.1 NFV MANO: Cross Domain

In order to manage the different domains, multiple NFV MANO instances, one for each domain, exist inside the deployment plane. Each NFV MANO can be decomposed into an NFV Orchestrator, a VNF manager, and a Virtual Infrastructure Manager (VIM); as per the ETSI NFV architecture. The NFV Orchestrator is responsible for the onboarding of Network Services (NS) and VNFs and for performing the lifecycle management of Network Services. It is also responsible for the validation and authorization of changes to the resources allocated to the VNFs. The VNF manager is responsible for the lifecycle management of one or a group of VNFs. Finally, the VIM is responsible for managing the infrastructure resources. Specifically, it manages the computational, network and storage resources.

3.3.2.2 WIM

The existence of different domains necessitates networks that connect/stitch all these domains. The component that is responsible for managing these networks is called the WIM, WAN - Wide Area Network - Infrastructure Manager. The WIM is a special case of a VIM. While the latter manages all of computational, storage and networking resources, the former is specialized in managing networks. Its main objective is to connect/stitch the different VNFs within a single domain or across several technological and/or administrative domains. The deployment plane can accommodate several WIMs that are used to “stitch” the different Network Services that are deployed in different domains.

3.3.2.3 Virtualized infrastructure

The virtualisation layer offers a unified view of the computational, storage and network resources. This unified view helps to aggregate and seamlessly run VNFs on top of the infrastructure. A Network Service can be composed of multiple VNFs, able to run either on Virtual Machines (VMs) or on containers. Up until recently, both VMs and containers could not be run on the same infrastructure. Recently, ETSI defined the Container Infrastructure Service Management (CISM) that enables MANO infrastructure to support containerized workloads, and thus, VMs and containers can coexist on the same infrastructure. The main responsibility of CISM is to manage the infrastructure’s resources and to perform the lifecycle management of the containers running on top of the container cluster. ETSI has proposed different architectures on how the container cluster can be implemented [83]. For instance, the containers can be run on bare metal, in VMs, or they can be distributed between the two.

3.3.2.4 Monitoring agents

Monitoring the running VNFs is of utmost importance. It is the foundation upon which the functionalities of the domain-specific monitoring and reaction plane as well as the functionalities of the E2E conducting plane highly depend. Indeed, in order to be able to predict service-level agreements (SLA)s violation, and to promptly react to and mitigate such events, a thorough monitoring of the infrastructure and the running software is mandatory. The monitored data can differ in location and in nature. The monitored data can be service level information that can be gathered from VNFs. It can be infrastructure level such as computational, storage and network data. It can be also data from the end users, such as the QoE. Thus, besides the VNFs, a myriad of monitoring agents are deployed across the infrastructure and at different levels.

3.3.2.5 XR device & XR service enabler controllers

The XR device controller is a piece of software that acts as a controller for the XR devices. This would allow to split the data plane from the control plane. Such a controller would allow the reconfiguration of the deployed XR devices and their synchronization with the XR services. Likewise, the XR service enabler controllers are targeted to control XR specific services instead of devices. Specific controllers or open sources for the control/remote configuration of XR devices and XR service enablers will be identified further during the course of the project, particularly as part of the research activities in WP3.

3.3.3 Domain-Specific XR Service Monitoring & Reaction Plane

The domain-specific monitoring and reaction plane is responsible for managing only one domain. Its main responsibilities are: i) keep track of the resources consumption and of the XR services running in the domain; ii) process the monitored data; iii) perform local analytics, make decisions and carry out the actuation which are specific to each running XR service.

3.3.3.1 Resource & XR service indexing

This plane keeps track of the domain resource usage and lists the domain capabilities and the domain running services. The advantages of recording this kind of information are manifold. For instance, a "service registry and discovery" entity would allow different tenants to reuse and share the same service. Similarly, prediction mechanisms can use the evolution of resource utilization to predict eventual SLA violations and/or service degradations.

3.3.3.2 Data collection, filtering and pre-processing

One of the responsibilities of this plane is the collection, filtering, and pre-processing of the monitored data. Data collection is done by agents located within the infrastructure, beside the VNFs. There are mainly two types of data collection agents, namely, the ones using the push method and the ones using the pull method. The former type resides beside the monitored entity and send the collected data to a server (i.e., a corresponding service or micro-service at the domain-specific monitoring and reaction plane), whilst the latter type sends the collected data only when instructed/inquired by the relevant service. Generally, both of these methods coexist in the same infrastructure. Filtering monitored data helps reducing the data size to be processed, by filtering out duplicates and unnecessary data. The level of filtering can be dynamically adjusted according to the needs of the monitored XR application. The pre-processing can have multiple forms. Its main purpose is to help other entities to seamlessly ingest the monitored data. For instance, it can be in the form of feature extraction algorithms that help reduce the feature number and the dimension of the data which, in turn, reduces the needed bandwidth for transferring the monitored data. It can be also in the form of converting the data into a specific format. Pre-processing can also consist of a distributed ML algorithm whereby the first few layers of the neural come network are calculated close to the data source.

3.3.3.3 The automation loop

Following the ZSM framework, this plane implements a local automation loop. This loop consists of the monitoring entity described above, that is shared among all XR services, an analytics engine, a decision engine, and an actuation engine, that are dedicated to each XR service. The analytics engine consists of algorithms that ingest the monitored data and then produce insights or alerts. For instance, such algorithms can predict the state of the service, detect misbehaving services and attacks, and identify optimizations that can greatly enhance the QoS. These algorithms are mostly designed using ML algorithms. The decision engine uses the insights gained from the analytics engine in order to derive its decisions. It can receive alerts from the analytics engine, as it can directly ingest low level monitored data. Its main purpose is to make sure that the running XR services keep meeting their SLAs. It can carry out decisions such as service re-composition, service migration, or even slight VNF re-configurations. Finally, the actuation engine's role is to decide how to implement the decisions that were made by the decision engine. Mainly, it is responsible for translating decisions into actions, and executing the produced actions within the relevant entities.

3.3.4 XR Service E2E Conducting Plane

The E2E conducting plane is responsible for managing the overall XR framework. It is the entry point for XR providers/developers from which they launch their services. It is also responsible for the lifecycle management of XR services.

3.3.4.1 XR service exposure

The XR service exposure is the gateway of the CHARITY platform for XR providers and developers. This gateway is used to create, setup, and manage XR services. XR providers and developers can also use this gateway to discover the services being offered by the CHARITY platform and by other third parties.

3.3.4.2 Resource registry and discovery

This plane keeps track of all the running XR services. It records, near real-time, information about the resource consumption of all domains. It also holds information about domain capabilities and generic XR service blueprints. New resources can be discovered and dynamically added to the CHARITY platform. They can also be released if they are not being used by the running XR services. Once resources are added to the platform, their statuses and their consumption rates are kept continuously updated.

3.3.4.3 Service planning and deployment

It is the main place where XR service planning is conducted. Upon receiving a request from an XR provider/developer, it translates the request into a blueprint, selects the set of domains where the service will be split upon, and carries out the negotiation with the respective domains. In order to carry successful deployments, this entity uses the XR Service Blueprint Template, XR Service Enabler, and Running XR services repositories. The XR Service Blueprint Template repository is used to store and manage the templates of XR services. The templates consist in the definition of many services and the interconnection between them. These services can be standalone VNFs or the description of other lower-level services. The XR Service Enabler repository contains how the services referenced in the XR service blueprints are implemented. For instance, if an XR service needs a video encoder service, this repository would contain many implementations of this encoder service, where each implementation has its own characteristics. The orchestration would select the best encoder according to the needs of the XR service and the current state of the platform. Finally, the Running XR Services repository is used to track the status of the running XR services. It contains all the information that relates to the running XR service, such as the health, the load, the uptime, etc.

3.3.4.4 E2E automation loop

Similar to the automation loop in Domain-specific monitoring and reaction plane, this plane also contains a loop. There is a loop for each XR service and it is responsible for E2E level service re-composition. The E2E analytics engine takes inputs from analytics engines of all domains where the XR service is running. Correlations between monitored data, produced from different domains, are used to produce alerts and state predictions of the XR service. The robustness of the E2E analytics engine algorithms is an important issue since this engine is responsible for monitoring E2E KPIs.

Even in cases whereby E2E KPIs are monitored by User Equipment (UE) for reliable detection of service degradation, the E2E analytics engine is responsible for finding the reason behind such service degradations. The role of the E2E decision engine is to decide on the course of action that needs to be taken in order to keep the XR service up and running. This means that the E2E decision engine can perform service re-composition per domain as it can perform an E2E level service re-composition. Such decisions may consist in VNFs reconfiguration, scale up and scale down operations, VNFs migration, or even domains re-selection. There are multiple reasons why service re-compositions may become required. For instance, they may be due to security concerns, to avoid a future service degradation or mitigate an existing one, or to optimize the resources utilization by the XR service. Finally, the E2E actuation engine is responsible for implementing the decisions made by the E2E decision engine.

3.3.4.5 CI/CD Pipeline

During the last few years, the DevOps concept is becoming highly popular, as there is a growing number of companies that are embracing that concept. DevOps brings down the separation between the development team and the production team, which results in a smaller time to market for new features and products. One of the main pillars of DevOps is automation, which consists in automating the build, the deployment and the monitoring of an application. Therefore, a CI/CD pipeline is needed in order to automate the build and deployment of XR services. This pipeline is used to integrate the service provided by an XR developer with the services provided by other developers or by the XR platform itself. Once the new version of the service is thoroughly tested, it can be deployed. For instance, the pipeline can perform a rolling update, where the newest version of the service is gradually deployed, as it can expand testing by performing a canary deployment, where only a fraction of the XR traffic is routed to the newest version of the application.

3.3.5 Application Management Framework (OSS/BSS)

Along with the provisioning of advanced XR Service enablers, CHARITY has the goal to make such enhanced capabilities as accessible and usable by XR Application Developers as possible. This brings the need for a component that can optimize the access to such capabilities, namely Application Management Framework (AMF). The services and tools incorporated in this component will support improvements to the XR application development cycle, in terms of speed, cost and effectiveness. The common CI/CD (Continuous Integration/Continuous Delivery) model, shown in Figure 41, is a baseline of this component, and will potentially tackle the integration and testing phases of the development cycle.

Figure 41: Generic CI/CD cycle

Part of this component is the design time management of network slice blueprints. The AMF framework will act as the main entry point for XR application developers to access the functions for defining and handling their XR services. So, the component is expected to include an external interface, whose specifics will be defined in the implementation phase.

Figure 42 depicts the CHARITY high level architecture, as proposed in the description of work, wherein the scope of the AMF framework and the CI/CD component are highlighted.

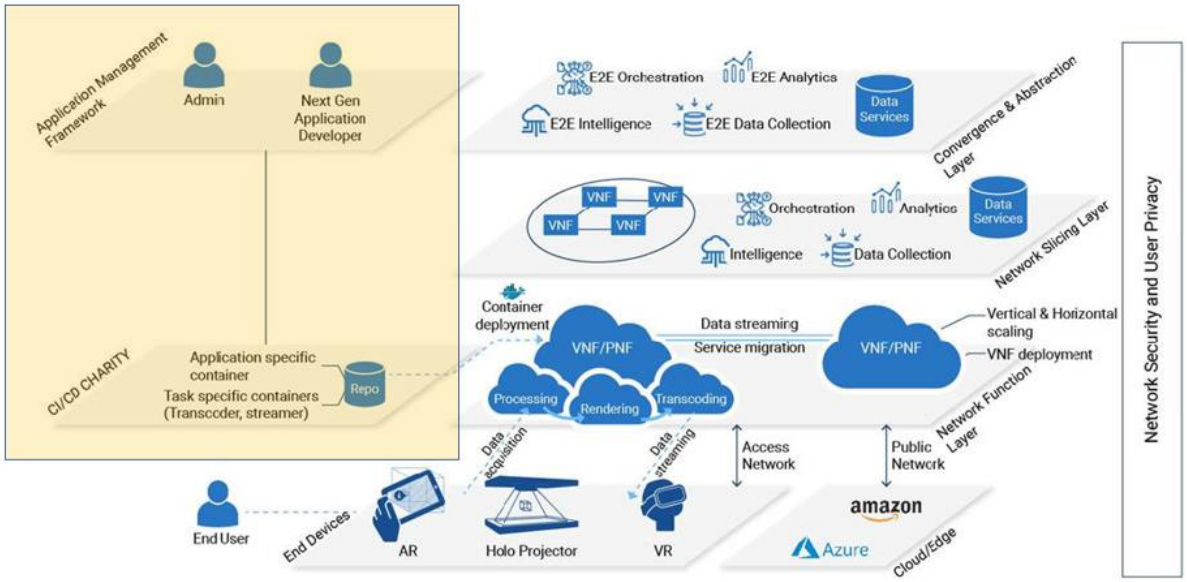

Figure 42: AMF place in the original CHARITY diagram - as submitted in the description of work

The internal interface of the AMF is towards the XR Service Orchestration layer. It is envisioned as a loosely coupled interface, including an asynchronous communication mechanism (that, consistently with the general CHARITY vision, might be a messaging, publish/subscribe interface), and some shared repositories, that will allow to share the artefacts generated by the AMF at design time with the Orchestration layer that has to use them for the application runtime deployment. The interface will include some kind of translation layer, whose purpose is to convert an abstract XR service description into a runtime description, including all the parameters requested by the Orchestrator to deploy the application and not abstractly definable at design time.

At the moment, the functions defined for the AMF component can be seen in two groups:

- Functions specifically supporting the design of network slice blueprints
- Other functions supporting the XR application composition and testing

Slice blueprint handling can in general include the following:

- Design of abstract network slices, intended as blends of Virtual Network Functions into Network Services, consistently with a model like the ETSI MANO (see section 2.2). The blend includes CHARITY provided artefacts (XR service enablers), virtual links, in some case PNFs (physical network functions) when making sense.
- Validation of generated network slice blueprints.
- Registration and storing in a common repository (XR Service Blueprint Templates Repository) of abstract network slice blueprints.
- Update of registered network slice blueprints.

Other functions can include:

- XR Application registration/onboarding and management through a CI/CD chain. This encompasses the uploading of application components in a repository shared with the Orchestration layer, accompanied by a proper abstract description. The components can in general be uploaded as source code, or as images already packaged into virtualised wrappings (virtual machines, containers, etc.).
- Definition of application model templates describing the different application bricks, along with the description of their interconnection and interoperation.

- Validation of composed applications in a segregated testing environment.
- Management of dynamic changes to the application model during the application execution, including updates to running microservices, addition of new microservices, or decommissioning of running ones, keeping the consistency with the abstract application model repositories.

The CI/CD pipeline incorporated in this component will be based on the typical models, where the stages actually useful to CHARITY will be selected and implemented. The pipe might include enhancements to create a DevSecOps pipeline, i.e., a cycle where security checks are additionally performed on the software, at static source code stage as well as in wrapped format and in dynamic mode. The proper parts of a DevSecOps cycle will be picked, and in the implementation phase it will be decided to embed them into the general CI/CD pipe, versus implementing them as separate self-standing tools.

3.4 Procedures for XR services deployment and management

This section presents the procedures of the XR platform that help in the deployment and maintenance of XR services.

In what follows, we consider the use case of a live XR concert using holography, which is a concrete implementation of the use case 1 of the CHARITY project [84]. This use case consists of two musicians, present in two different cities, performing a live show that is projected in three venues, a music hall, a stadium and a park. Given the ultra-low latency requirements for music and the extremely high bandwidth requirement for holography, two network links would connect each musician to the cloud. Due to the fact that the synchronization of music happens in the cloud, links with deterministic latency should be used to carry the sound from the musicians to the cloud. For the holography links, some pre-processing can take place in the edge, close to the musicians, and the rest of the processing takes place in the cloud where it is also synchronized along with the audio. Fortunately, due to human perception, the synchronization between audio and video is not very restrictive which explains why the links transporting the videos are not deterministic links. Finally, the resulting stream is sent to the three venues, where it is decoded, adapted to the holography devices and projected. Given the nature of the venues, the required resources for each venue may differ. Indeed, the stadium and the concert hall can have a wired connection with processing power that belongs to the XR provider acting as an edge, while the park venue would use wireless technology coupled with edge providers. Other use cases are expected to follow a similar deployment model.

3.4.1 Launching an XR service

The instantiation of an XR service follows the procedure illustrated in Figure 43. An XR provider can be contacted by an end user in order to launch an XR service. The XR provider sends a request to the XR platform detailing the service to be launched. The XR platform or the E2E conducting plane checks if the service can be launched in the current state of the platform. This feasibility check makes sure that the platform has enough resources to accommodate the new service. Using a Blueprint Template Repository and an Enabler Repository, the request of the XR provider is translated into one or multiple detailed blueprints characterizing the XR service to be deployed. The XR platform checks the feasibility and cost of the blueprints and selects one of them to be deployed. Once a detailed blueprint is selected, the domains that will host the XR service are selected. To launch services inside the domains, the E2E conducting plane uses an entity dubbed Cross- Domain Resource MANO that offers a unified interface to all administrative and technological domains composing the XR platform. This interface first instantiates the required services for the E2E automation loop, and then initiates the automation loops of the new XR service within each domain and connects them to the E2E automation loop. For the MS, this can consist in setting up a monitoring system (e.g., Prometheus) that is specific for the XR service. For the AE and DE, it can consist in connecting them to all the relevant monitoring systems and also publishing their capabilities so they can be used by other services. The automation loops can be

considered as a platform that can hold the algorithms for AE and DE. These algorithms can be dynamically added and removed at runtime. For instance, in the AE platform, it is possible to have many ML models for different purposes whereby they can be replaced by newer versions when needed. While in DEs, it is possible to have different ML models running in parallel voting on what would be the best decision, some of these algorithms may be proposed by the XR platform, XR providers, or even by third parties. Finally, the E2E conducting plane launches and configures the Service Function Chains (SFCs) inside each domain, and uses the WIM to stitch the different SFCs across the different domains. The VNFs composing the SFCs should have monitoring agents that report to the domain-specific automation loop. Once the E2E XR service is up and running, the E2E conducting plane informs the XR provider that the XR service is ready, and ultimately the end-user is accordingly notified.

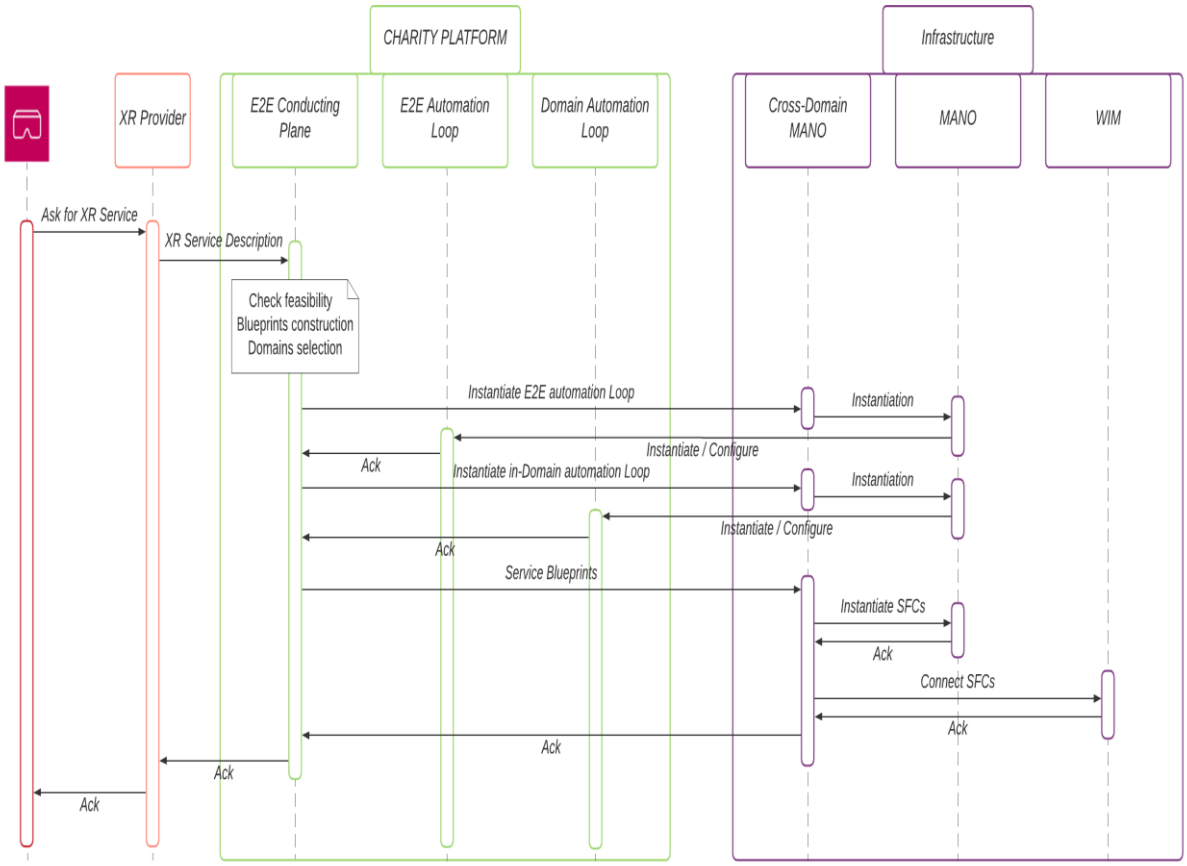

Figure 43: Launching an XR service

3.4.2 Modification of an XR service

Figure 44 depicts the procedures to modify slices in order to preserve the good functioning of an XR service. When an analytics engine notices or predicts a significant degradation in QoS/QoE in the near future, it sends alerts to the decision engine of its own domain. If the domain-specific decision engine can mitigate the issue inside the domain, it sends its decision to the respective actuator that will implement the decision within the boundaries of the domain. If the domain specific decision engine cannot resolve the issue at the domain level, it sends a modification request to the E2E decision engine. The E2E decision engine may also receive alerts directly from the E2E analytics engine. After receiving an alert or a modification request, the E2E decision engine can decide to gather more insights and search for a new service re-composition that will keep the XR service healthy. Once a feasible configuration is found, it sends it to the E2E actuator.

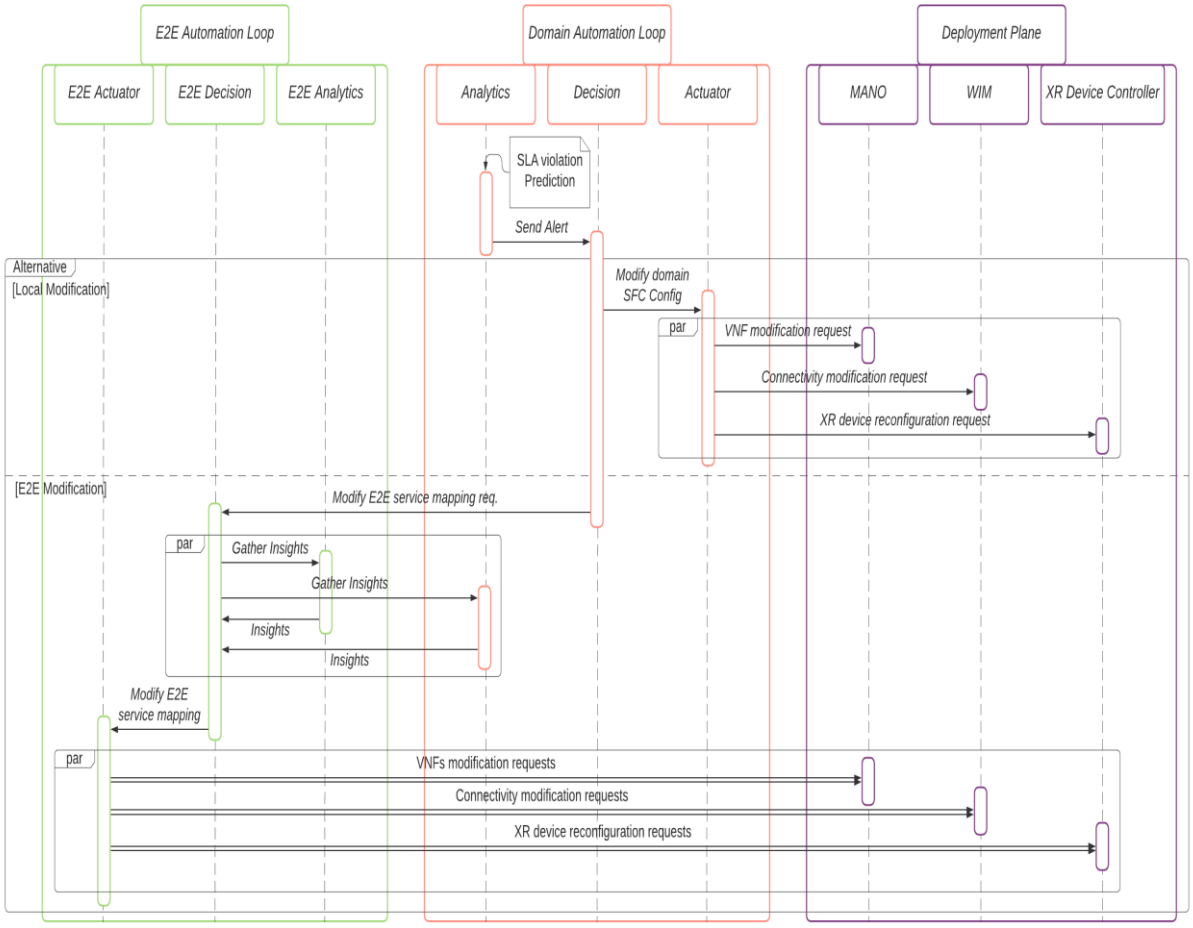

Figure 44: Modify an XR service

3.5 Tools and Open Sources

This section demonstrates the feasibility of the proposed architecture by mapping its functionalities to existing tools and open sources.

Recently, we have witnessed an explosion of tools and open-source components focused on service, network, and infrastructure orchestration. ETSI MANO aligned implementations, such as Open-Source MANO (OSM), ONAP, OpenBaton, or OPNFV, have been under active development in the last few years and are increasingly mature (and complex). Driven by the telecommunication sector, such MANO solutions leverage the advances of virtualisation and containerization technology to create a more cost-efficient and flexible approach for the deployment and management of VNFs and NSs on top of commodity hardware.

For instance, OSM, one of the most popular community-driven solutions, is an ETSI-hosted project used to model and orchestrate network services with the support for different VIMs. In the proposed XR platform, these MANO tools can fit the XR Service Deployment Plane where distinct domains (and implementations), both technological and administrative, might coexist. Moreover, this multi-domain vision requires an integrated cross-domain resource component such as an actual ETSI MANO instance with multi-domain support or an additional tool such as Apache Libcloud or Terraform. The cross-domain resource component allows abstracting the different infrastructure providers APIs and provides the required interoperability across distinct underlying environments. For instance, Apache Libcloud allows interacting with multiple popular cloud providers using a unified API, whilst Terraform can be used to reassemble a single workflow to efficiently manage infrastructure and specify different component blueprints (i.e. compute instances, storage, network) within distinct service providers. Additional tools such as the Openslice can also have an important role into the onboarding and management of NSs and VNFs.

On the other hand, the implementation and management of a virtualisation infrastructure is a great challenge. It requires a comprehensive approach to stitch together the ephemeral and distributed large number of micro-services. Multi-domain use cases require such micro-services to be deployed and located across multiple infrastructure providers. In the same way, the wide spread of diverse edge/cloud environments impose different challenges pertaining to the automation and optimization of the overall orchestration process, the needed observability over the infrastructure (both north-south and east-west network traffic), and the security and privacy enforcement of the XR services across these hybrid edge/cloud environments.

In that regard, Kubernetes have become a de facto platform to support the orchestration of micro-services within a Cloud-Native environment. Kubernetes, a Cloud Native Computing Foundation (CNCF) graduated project and supported by all the major cloud providers, is today a popular choice for the deployment, automation and management of container-based services and applications. Indeed, many distinct Kubernetes distributions (and platforms built on the top of Kubernetes) exist like OpenShift, AWS Elastic Kubernetes Service, Google Kubernetes Engine, Azure Kubernetes Service or even Kubernetes-based platforms designed with lightweight environments in mind such as Rancher, K3S, Microk8s, or KubeEdge. The later ones are more focused on resource-constrained deployments such as local or edge domains. Thus, in the proposed XR platform, Kubernetes can be explored to support the container-based workloads of the envisioned next-generation applications. Additionally, tools such as KubeVirt can be also considered to address the inevitable transitioning from classical Virtual Machine-based workloads through a common platform.

The myriad of integration approaches and technology options generates a side effect which results in highly complex and heterogeneous environments, especially when considering edge/cloud environments, as mentioned before. Within the proposed XR platform, this means the possibility to provide not only the requiring stitching between the provisioned XR services but also an efficient integration and communication among all the components of all planes and domains, including the Domain-Specific XR Service Monitoring and Reaction and the XR Service E2E Conducting planes.

For Cloud-Native environments, one feasible path is to leverage the concept of Service Mesh along with different integration fabrics (i.e., the domain and cross-domain integration fabrics and the Conductor Integration Fabric). The underlying Service Mesh concept relies on the usage of network proxies (the so-called sidecars), together with each service (or using different arrangements such as one proxy per node). Then, those network proxies allow to observe, control and implement security features across all the network traffic between components. While this provides a more unified approach to manage network communications within a Cloud-Native environment, it also brings additional latency (i.e., the network traffic needs to go through the proxies) and complexity (i.e., they also need orchestration). Nevertheless, the service mesh and the Integration Fabrics concepts might provide a more flexible communication strategy to support the network communications among all the XR platform elements.

Numerous Service Mesh implementations have emerged in the past years like Istio, Network Service Mesh, Linkerd, Consul, Traefik Mesh, Open Service Mesh or GlooMesh. Most of these Service Mesh implementations rely on the Envoy sidecar implementation to realize the Service Mesh concept. Still, some of them address different use cases like multi-cluster management or support different network protocols. For instance, Istio, one of the most widely used Service Mesh implementations, works at layers 4-7. Amongst others, Istio provides service discovery, traffic management capabilities, traffic encryption between services, observability over the communications, and built-in access control mechanisms. Moreover, Istio and Envoy can also delegate the access control to external policy enforcement tools such as the Open Policy Agent. Rather than network focused, such an approach might turn into a more comprehensive approach for applying different policies across distinct components of the XR platform by leveraging a common policy-driven strategy and syntax. On the other hand, Network Service Mesh, another Service Mesh implementation, working at layers 2-3, is a more recent attempt to support an additional range of use cases and network protocols.

Moreover, given the ever-growing Service Mesh ecosystem, a standard Service Mesh interface was proposed to unify the consumption of Service Meshes APIs across different implementations. Such a standard interface plays a relevant role to allow multi-domain setups, as discussed before. In the same way, specific tools, such as the GlooMesh, also address the problem of integrating multiple and different domains through a single management interface on top of already existing distinct Service Mesh environments. On the other hand, given the additional hop in the network communications, the Service Mesh concept implies a performance penalty in the already strict requirements of an XR service. Therefore, aside from the functionalities, the efficiency of different Service Mesh implementations needs to be considered for the full realization of the proposed XR platform.

The monitoring and reaction plane and the notion of closed automated loops are critical characteristics that shall be part of the XR platform. These loops, as discussed before, rely on the ability to collect and process different kinds of data as input for the analytics logic. Such data collection, which shall occur near real-time, is essential to understand the services provided by the XR platform, the network communications and the infrastructure (e.g., edge/cloud resources) where the various components run. The Service Mesh concept can also support such monitoring by providing service and network related insights to the monitoring and reaction plane. Additional monitoring related tools, such as Prometheus, can be used to gather insights from the different components in a more comprehensive way. Prometheus is a widely used monitoring solution for collecting and storing metrics from third-party systems, supported by a growing number of plugins. In the proposed XR platform, Prometheus can be leveraged and integrated with the service mesh sidecars up to the integration fabric components. Other solutions, such as the ElasticSearch stack, more focused on logs can also fit the overall monitoring approach.

Likewise, for the feasibility of the integration fabric concept (which comprises the messaging bus), multiple tools and open-source components like Apache Kafka, RabbitMQ or ActiveMQ, amongst many others, can be leveraged. For instance, Apache Kafka, one of the most widely used messaging systems, has the notion of a distributed set of messaging brokers organized within elastic clusters. Kafka brokers are used to storing and serving messages. Those messages, organized by topics and partitions, can then be partitioned and replicated, respectively, ensuring different service levels of performance, durability, or fault tolerance, and that is according to the needs of each use case. In the proposed XR platform, Kafka has a decisive role in providing asynchronous bus communication channels to interconnect the elements within each domain and all the XR platform domains. In this way, Kafka messages can be used to model all the communication functionalities (e.g., service registration, discovery, and monitoring messages) and facilitate the integration with external systems. As a reference example, OSM follows a similar approach, whereby Kafka has the role of messaging bus for OSM components.

Figure 45 depicts the initial map between the most relevant surveyed open sources and tools with the key components of the CHARITY architecture. In the initial stage, we focused on understanding existing open sources components and tools to fulfil the vision of CHARITY as a distributed platform across multiple domains. Namely, we targeted tools to expose and instantiate resources spanning distinct domains (i.e., cross-domain resource MANO tools), how to monitor them (i.e., service-meshes and their related monitoring components) and finally, how to interconnect CHARITY components through efficient integration fabrics, both within and across domains.

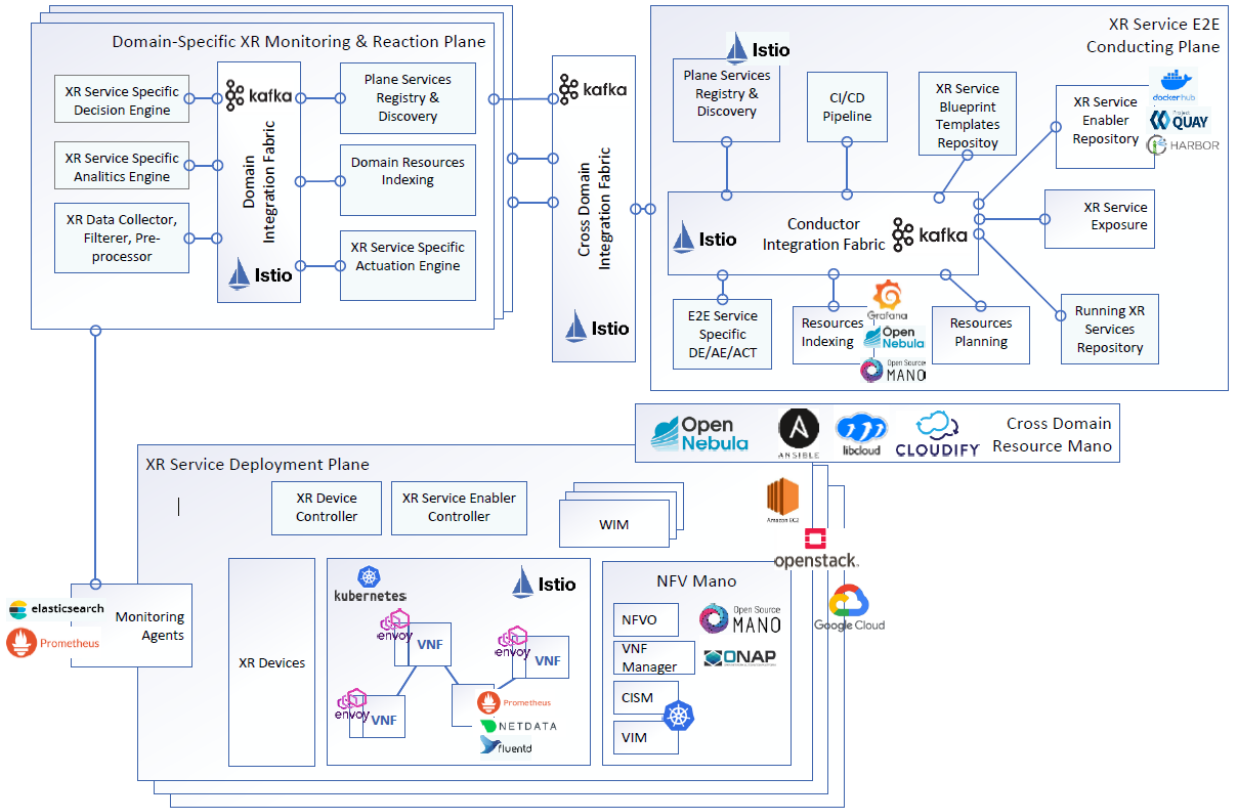

Figure 45: Mapping of candidate open sources and tools with the CHARITY framework

4 CHARITY Architecture – Use Cases & Stakeholders

In this section, we validate the CHARITY architecture, as introduced in Section 3, against the envisioned use cases of CHARITY. The relevant stakeholders will be also identified. Potential business models are outside the scope of this deliverable, and are to be detailed in the relevant deliverables of WP5.

4.1 Stakeholders

In this section, we introduce the main stakeholders according to each of the Use Cases.

Definition of a stakeholder: In the context of the CHARITY project, and as a general definition, stakeholders are considered any entity that has an invested interest in the tangible outcomes of the project. This includes both academic and industry partners committing expertise and resources to the project, as well as internal and external organisations making use of the project outcomes.

Essentially the stakeholders can be divided into two groups; providers and consumers (i.e., intuitively in addition to regulatory/government bodies). Having a better understanding of the stakeholders at the provider end will help us to make more informative decisions when defining the general architecture outlined in the previous section, as well as the user-facing services in interfaces. Defining who the stakeholders are at the consumer end informs WP5, leading to a more targeted communication and dissemination of results. Stakeholder mapping is an important step in creating value throughout the CHARITY project and for identifying the most relevant target audiences for the communication and dissemination strategy undertaken in WP5.

The stakeholder mapping further identifies four sub-categories of stakeholders: 1. core stakeholders, consisting of (Service providers and Vendors), 2. external partners (Cloud providers, Network operators, Network equipment vendors, XR Equipment vendors), 3. recipients of the project outcomes (Researchers), as well as 4. End Users / Customers.

4.1.1 Use Case 1: Real-time Holographic applications

The stakeholders of the Real-time Holographic Applications use case include the following:

Holographic Meeting scenario:

- Application Developer/Provider: Holo3D – Avcom Entertainment SRL
- XR Application: Holographic Application
- CHARITY platform: CHARITY XR platform
- CHARITY: Project consortium
- Cloud/Edge Provider: Provider of edge and cloud resources (providing GPU accelerated VM-services)
- Holographic display vendor
- Client: Event host and/or moderator
- End-users: Speaker (PC with camera & microphone) and Audience

The stakeholders listed above are relevant to the Holographic Meeting scenario presented by Holo3D. In this scenario, the **Speaker** is recorded (audio/video) during a meeting or conference. The speaker can be located anywhere assuming they have access to a high-speed internet connection. The resulting video is streamed live to the **CHARITY platform** which is hosted by a **Cloud/Edge Provider**. The **Holographic Application** then “scrambles” the 2D video in real-time and adapts it to 3D. The **Client** receives the stream via the CHARITY Platform and it is projected using **Holographic Displays** to the **Audience** (end-user).

It should be noted that communication is in real-time and that the solution works in both directions, allowing for two way conversations between the speaker and the audience. There can be multiple

speakers and the meeting can be streamed to multiple Holographic Displays in different geographical locations.

Live Holographic Music Performance scenario:

- Application Developer/Provider: Holo3D – Avcom Entertainment SRL
- CHARITY platform: CHARITY XR platform
- CHARITY: project consortium
- Cloud/Edge Provider: Provider of edge and cloud resources (providing GPU accelerated VM-services)
- Holographic display vendor
- Client: music venue
- End-users: Musician(s)/Performer(s) (PC with camera & microphone) and Audience

For this scenario the stakeholders can be considered the same as in scenario #1. However, this time the Speaker/s are the band members performing live music to an audience. The audience is filmed and streamed back to the performers in real-time. However, this time the audience does not interact on a one-to-one basis.

Holographic Assistant Application scenario:

- Application Developer / Provider (SRT)
- SRT APP: Holographic Assistant application
- CHARITY XR platform: the platform based on CHARITY architecture supporting XR services
- CHARITY Admin: XR platform provider
- Cloud/Edge Provider: Provider of edge and cloud resources (providing GPU accelerated VM-services)
- SRT Client: local system, holographic 3D display device and hardware incl. Eye-Tracking
- End-user: user of SRT APP, watching the hologram
- Internet Service Provider: Information services provider, e.g. text2speech, speech2text, weather data, stocks, chat bot

For this use case the **Application Developer** creates the application providing the Holographic Assistant and its features as like as the software modules to compress, decompress and transfer 3D point-could data from cloud to local **SRT Client**. On **SRT Client** side a local application presents the Holographic Assistant content on a holographic 3D display. The cloud part of the application is exposed via the CHARITY platform which is in turn hosted on a **Cloud/Edge Provider**. The Holographic assistant may directly or indirectly use **Internet Service Provider** to show information or content requested by the **End-User**.

4.1.2 Use Case 2: Immersive virtual training

The stakeholders of the Immersive Virtual Training use case include the following:

VR Medical Training Application scenario:

- Application Developer/Provider: ORAMA
- Application: ORAMA-VR
- CHARITY platform: XR platform
- Cloud/Edge Provider: Provider of edge and cloud resources
- VR equipment vendors (PC, and/or Head-mounted display, other controllers)
- Customer: University - Company
- End-user: Trainee

For this use case the **Application Developer** creates the content, authoring and training module as part of the application. The application is exposed via the **CHARITY platform** which is in turn hosted on a **Cloud/Edge Provider**. The **End-user(s)** connect to the virtual training course using a head-mounted display (HMD) and any other controllers necessary for the training module. It should be noted that the **Customer** in this case could be a university or a company that provides the training products to members internally.

VR Tour Application Scenario:

- Application Developer/Provider: DOTESFERA
- UC owner-APP: Cyango
- XR Front-end platform: Story and Cloud Studio
- CHARITY: XR platform
- CHARITY-Admin: XR-platform provider. It is responsible for maintaining CHARITY (e.g., developing some enablers such as PC-ENC/DEC, business agreements with public cloud/edge providers, etc.)
- Customers / End-users: Marketing Agencies and Creators
- Cloud/Edge Provider: Providers of edge and cloud resources
- VR equipment vendors: Oculus

The **Application Developer** (in this case DOTES) deploys the **Cyango Virtual Tours software** to a powerful media server in the **Cloud**. This is required for the file hosting and real-time image processing so the **User** can start to create a virtual tour or to build interactive experiences in virtual reality. While the host can use a mobile device to capture the video and audio, the editing is done in the cloud. In this way, the host can live stream from his camera and the server delivers the output with real time translations for text and audio to the **Viewer**.

4.1.3 Use Case 3: Mixed Reality interactive application

The stakeholders of the Mixed Reality interactive application use case includes the following:

Collaborative Gaming scenario:

- Application Developer/Provider: ORBK
- XR Application: AR Collaborative Gaming
- CHARITY platform: XR platform
- Game Server
- Cloud/Edge Provider: Provider of edge and cloud resources
- Client: mobile device
- End-User: Player, a person playing the game

The **Application Developer** (ORBK) develops a highly immersive multiplayer AR game along with a dedicated multiplayer engine for synchronizing the dynamic game objects and user states across all end devices. The **CHARITY platform** provides the architecture allowing the application developer to deploy the game to the **Game Server** which is hosted in the **Cloud**. The **Player** starts the game application on their end-device and either joins an existing game (hosted on a nearby game server) or creates a new game server which is then deployed on a hosted machine close to the player. The player can then scan the environment to gather 3D point data required for the mesh collider generator which is sent to the game server.

Manned-Unmanned Operations Trainer Application scenario:

- Application Developer: Collins
- XR Application: Collins-APP

- Cloud/Edge Provider: (Hosts the required software components; Flight Oracle, Arena Management)
- CHARITY: XR platform
- Cloud/Edge Provider: Provider of edge and cloud resources
- XR equipment vendors: PC, and/or Head-mounted display, other controllers
- End-Users: Trainees

The **Trainee(s)** login to Collins-APP and establish a session. Each trainee registers with the **Arena Manager,** hosted in the **Cloud** to synchronize positions with other Trainees (in the case of interactive training). Trainees are presented with true immersion with real-world cockpit controls, and generated scenery being generated outside the cockpit windows. With the **User Device** the trainee can interact with the cockpit controls. The **Flight Oracle** component**,**hosted in the cloud, predicts the pre-fetches rendered terrain images from the cloud to be rendered both in the cloud and at the edge. Background scenery is rendered in the cloud since it is more resource intensive, while closer objects are rendered on the edge. The **Arena Management** component is also hosted in the cloud to keep track of all other aircraft deployed within the simulation and relays this to the **Scene Management** components hosted on the **Edge**. **Session Agents** are deployed on the edge node selected by the **CHARITY Orchestrator** and act as the sole point of contact for components running on the **User Device.**

4.2 Use Cases – Walkthrough

The CHARITY project envisions the creation of a Cloud framework that will host demanding applications for Virtual Reality, Augmented Reality, and Holography by utilizing edge resources and devices. The CHARITY platform will be designed upon and tested through a set of representative UCs, owned by partners of the consortium. These UCs, seven in total, are organized in three main categories, namely a) Real-time Holographic Applications, b) Immersive Virtual Training and c) Mixed Reality Interactive Applications. Table 2 encapsulates this categorization along with the names of the respective UC owners.

Table 2. List of UCs with the respective owners in CHARITY

UC1: Real-time Holographic applications	
UC1-1: Holographic Concerts	HOLO3D
UC1-2: Holographic Meetings	HOLO3D
UC1-3: Holographic Assistant	SRT
UC2: Immersive Virtual Training	
UC2-1: VR Medical Training	ORAMA
UC2-2: VR Tour Creator	DOTES
UC3: Mixed Reality Interactive applications	
UC3-1: Collaborative Gaming	ORBK
UC3-2: Manned-Unmanned Operations Trainer	UTRC

The CHARITY project aims to impact these three use case areas by providing a platform tailored to accommodate relevant applications and suitable to support their innovative features. CHARITY expects to deliver a working prototype that will meet the requirements of such applications and ultimately pave the way to the mass adoption of more advanced media applications.

The following sections will describe how the UCs would take advantage of the proposed architecture. Indeed, it will be shown how the component of each use case would fit within the architecture. Each of the UCs is organized as follows: i) A brief introduction to the use case. ii) An application development phase where the details about the development of the XR application are given. iii) The application deployment phase shows how the XR application is deployed on top of the CHARITY platform. iv) The application exploitation phase shows the interplay between CHARITY platform and the XR application in order to meet the KPIs and preserve the QoE/QoS of the XR application. v) The application decommissioning phase shows how the decommission and reclaiming of resources are done.

4.2.1 UC1-1 – Holographic Concerts

This UC uses a 3D holographic display that creates seemingly 3D holograms from transformed 2D video streams.

The Holographic Concerts use case is based on cloud processing supported by CHARITY services. Each client-PC (the PC located at the concert venue, that feeds the video content to the Holographic device) is assigned to its own instance of processing modules in the CHARITY platform. The use case is based on software and hardware supported by CHARITY cloud services. Some modules and services will run in the CHARITY cloud - i.e., 2D to 3D real time video transformation, compression and rendering. Some modules and services will run locally in the Client PC – i.e., Synchronization service.

The Musicians are filmed and their live streams are sent via HTTP to the CHARITY cloud service where it is transformed and streamed to the dedicated holographic display. CHARITY Cloud Service will be able to perform geometric transformations to the video stream (e.g., crop, resize, flip, etc.), in order to prepare it for the assigned type of holographic device. The PC in the venue location receives and synchronizes the video stream with the other Musician streams and displays it in the holographic display through HDMI interface. The Audience is also filmed, and their video stream is transmitted directly to the Musicians for feedback purposes from the stage. This reverse process is simpler since it does not require any video transformations or synchronization.

The CHARITY service should be able to perform simple video geometric transformations: scale, position, rotation, flip and crop. These transformations are required to make the video stream compatible with the assigned holographic display.

4.2.1.1 Application Development

During this phase, HOLO3D implements a new Holographic Concert application, directly following the given structure, the CHARITY XR platform offers.

The Holographic Concert application will consist of multiple software components running at different locations – local, edge and cloud. Such components consist in compression and decompression software and also multiple Video Content Manipulation Apps that would adapt the video traffic to different holographic devices. To achieve relevant KPIs and maintain QoS / QoE, several monitoring functions and control mechanisms will be implemented at the various modules to react to changing network conditions or requirements. Examples are adaptable compression parameters for such as variable resolution, colour bit depth or frame-rate. These mechanisms will be supported by the XR service monitoring & reaction plane, where appropriate services for analysis and decision making will be implemented. All communication between components will be based on TCP/IP.

At the end of this phase, all components and VMs needed for HOLO3D's Holographic Concert will be uploaded to 'XR Service Enabler Repository'. HOLO3D will also define a blueprint that describes how all the components relate to each other. This resulting blueprint, containing the description of all necessary functional & non-functional components as well as the topology of different components, will be uploaded to the 'XR Service Blueprint Template Repository'.

4.2.1.2 Application Deployment

In this phase, HOLO3D will deploy its application (Holographic Concert) on the CHARITY platform. This consists into selecting a blueprint from the 'XR Service Blueprint Template Repository' and customizing it before sending it to CHARITY for deployment. Once CHARITY receives the blueprint, it will use the 'XR Service Enabler Repository', 'Resource Planning', 'Resource Indexing', and 'Plane Services Registry & Discovery' entities in order to schedule the XR service. According to the information of the Musician and the chosen enablers, CHARITY will search and select the optimal domains where the different components of the XR service will be deployed. Each deployed component instance will be assigned to a musician. At this stage, some components need to be ran in the cloud or in the edge while the monitoring server would be deployed in the cloud.

The CHARITY orchestrator will deploy the components at the selected optimal domains and stitch these domains together to offer seamless connectivity between the components. Upon creation, the different components will register themselves in the 'Plane Services Registry & Discovery'. The registered components will use this plane to connect to each other. For instance, all monitoring agents will use that mechanism to find the monitoring server to which they will send the monitored data.

Once all components are up and running, the Client can begin using the Holographic Concert application. Indeed, the Musician can connect to the software components in edge and cloud. It is worth mentioning that by this time, the blueprint of the XR service will already be added to 'Running XR Services Repository'.

4.2.1.3 Application Exploitation

At this phase, the Holographic Concert application would be running, while the Audience would watch the result on the Holographic Device. During this phase, CHARITY would be responsible for ensuring the satisfaction of the KPIs, meaning that all closed loops would be functional. In what follows, we will consider some hypothetical scenarios to show the interplay between the different components.

- The E2E Service Specific AE would predict that the bandwidth between cloud and venue would not be enough during the next few minutes. In order to preserve the KPIs, the DE would choose to alleviate this issue by choosing and adapting suitable parameters for compression
- If the AE predicts that this perturbation would persist for a long period of time, it alerts the DE, which will then decide to perform a migration of the relevant components. The application would choose the appropriate time for this migration (e.g., between two songs).
- Communication with a certain component stops unexpectedly. The XR Service Specific Data AE will detect this and alerts the DE. Fortunately, Given the sensitivity of this application, CHARITY platform would have been maintaining a fallback XR service with lower quality that would be used for similar situations. The DE would immediately switch to the lower quality stream and will provide a new instance of the missing component on a different location. Once the new instance is up and running the application would switch back to the high quality stream.

Obviously, it is the responsibility of the Actuation Engines to carry out and enforce the decisions made by the different DEs. Also, it is worth mentioning that all taken actions are validated by ensuring that they fall within the resource range available for HOLO3D Holographic Concert application.

4.2.1.4 Application Decommissioning

Finally, during this phase, the resources are reclaimed. The termination actions provided by HOLO3D Holographic Concert application are executed first; they are part of the XR service blueprint. The blueprint inside the 'Running XR Services Repository' is marked as 'terminating' and it is deleted once all the related components are decommissioned. Once the Holographic Concert application is terminated, the related shared services are inspected to be also decommissioned. If the shared services are not being used by other XR services, they are terminated. Otherwise, they will be kept,

but the closed loops inside them will eventually reduce the amount of committed resources to those services.

4.2.2 UC1-2 – Holographic Meetings

The objective of this cloud application is to enable the Speaker- the meeting participant that is filmed and their live streams is sent via HTTP to the CHARITY cloud service where it is transformed and streamed to the dedicated holographic display to be situated at any geolocation and transmit its video to numerous holographic displays in various venues concurrently. In this case, the Speaker-PC captures and transmits the 2D video to a CHARITY service that transforms and renders it accordingly, for any type holographic display connected to CHARITY. The Speaker-PC may receive un-edited video feed, filmed at the audience geolocation, to enable visual communication between the Speaker and the Audience-the meeting participants that see the hologram in real-time. Communication is therefore possible between various Speakers in the same venue. Furthermore, the Speaker will be able to share visual content with the audience.

Holographic Meetings use case is based on local processing supported by CHARITY services. Each client-PC is assigned to its own instance of processing modules in the CHARITY platform. The use case is based on software and hardware supported by CHARITY cloud services. Some modules and services will run in the CHARITY cloud - i.e., 2D to 3D real-time video transformation, compression and rendering. Some modules and services will run locally in the Speaker PC – i.e., shared content conversion service.

4.2.2.1 Application Development

During this phase, HOLO3D implements a new Holographic Meeting application, directly following the given structure, the CHARITY XR platform offers.

The Holographic Meeting application will consist of multiple software components running at different locations – local, edge and cloud. Such components consist in compression and decompression software and also multiple Video Content Manipulation Apps that would adapt the video traffic to different holographic devices. To achieve relevant KPIs and maintain QoS / QoE, several monitoring functions and control mechanisms will be implemented at the various modules to react to changing network conditions or requirements. Examples are adaptable compression parameters for such as variable resolution, colour bit depth or frame-rate. These mechanisms will be supported by the XR service monitoring & reaction plane, where appropriate services for analysis and decision making will be implemented. All communication between components will be based on TCP/IP.

At the end of this phase, all components and VMs needed for HOLO3D's Holographic Meeting will be uploaded to 'XR Service Enabler Repository'. HOLO3D will also define a blueprint that describes how all the components relate to each other. This resulting blueprint, containing the description of all necessary functional & non-functional components as well as the topology of different components, will be uploaded to the 'XR Service Blueprint Template Repository'.

4.2.2.2 Application Deployment

In this phase, HOLO3D will deploy its application (Holographic Meeting) on the CHARITY platform. This consists into selecting a blueprint from the 'XR Service Blueprint Template Repository' and customize it before sending it to CHARITY for deployment. Once CHARITY receives the blueprint, it will use the 'XR Service Enabler Repository', 'Resource Planning', 'Resource Indexing', and 'Plane Services Registry & Discovery' entities in order to schedule the XR service. According to the information of the Speaker, Audience and the chosen enablers, CHARITY will search and select the optimal domains where the different components of the XR service will be deployed. Each deployed component instance will be assigned to a speaker. At this stage, some components need to be ran in the cloud or edge while the monitoring server as like as the required micro-services would be deployed in the cloud.

The CHARITY orchestrator will deploy the components at the selected optimal domains and stitch these domains together to offer seamless connectivity between the components. Upon creation, the different components will register themselves in the 'Plane Services Registry & Discovery'. The registered components will use this plane to connect to each other. For instance, all monitoring agents will use that mechanism to find the monitoring server to which they will be sending the monitored data.

Once all components are up and running, the Client can begin using the Holographic Meeting application. Indeed, the Speaker can connect to the software components in edge and cloud. It is worth mentioning that by this time, the blueprint of the XR service will already be added to 'Running XR Services Repository'.

4.2.2.3 Application Exploitation

At this phase, the Holographic Meeting application would be running, while the Audience would watch the result on a holographic display. During this phase, CHARITY would be responsible for ensuring the satisfaction of the KPIs, meaning that all closed loops would be functional. In what follows, we will consider some hypothetical scenarios to show the interplay between the different components.

- The frame rate of received stream falls below given limit for a certain amount of time. Monitoring agents will continuously gather information about current frame rate, E2E latency and data rate. The AE will use these data to generate alerts that are sent to the DE. After analysis the XR service DE will decide on adapting and choosing suitable parameters for video compression. When the situation will be normalized when network conditions relax.
- Some component becomes unresponsive. The CHARITY platform will immediately provide a new instance of the missing component on a different location.
- Given the dynamic nature of this application, the number of venues can increase during the same session. In this case, the E2E Service Specific AE would predict that the bandwidth would not be enough to accommodate all the venues at once. So the E2E Service Specific DE would decide on splitting the load by deploying the same set of components in a new cloud location and mirror the source stream to the two sites. Each site would serve half of the 3d devices and thus the bandwidth would be split between two independent links.

Obviously, it is the responsibility of the Actuation Engines to carry out and enforce the decisions made by the different DEs. Also, it is worth mentioning that all taken actions are validated by ensuring that they fall within the resource range available for HOLO3D Holographic Meeting application.

4.2.2.4 Application Decommissioning

Finally, during this phase, the resources are reclaimed. The termination actions provided by HOLO3D Holographic Meeting application are executed first; they are part of the XR service blueprint. The blueprint inside the 'Running XR Services Repository' is marked as 'terminating' and it is deleted once all the related components are decommissioned. Once the Holographic Meeting application is terminated, the related shared services are inspected to be also decommissioned. If the shared services are not being used by other XR services, they are terminated. Otherwise, they will be kept, but the closed loops inside them will eventually reduce the amount of committed resources to those services.

4.2.3 UC1-3 – Holographic Assistant

The Holographic Assistant use case is based on software and hardware supported by CHARITY cloud services. Some modules and services will run in the CHARITY cloud - i.e., rendering, assistant logic and 3D point cloud processing. The assistant software thus runs independently from capabilities and resources of the end user device. Information and services offered by the holographic assistant are accessed through cloud-based services via APIs and interfaces provided by various data services

available on the internet (i.e., weather data, stocks, etc.). The advantage of this architecture is that the end user device only needs to visualize the resulting 3D data received as a streamed 3D point cloud. All gathering, processing of information, rendering, and compression of 3D data are performed in the cloud. Bandwidth and computation requirements are relatively low since the only data that needs to be transferred are the processed results and the assistant visual/audio information (especially important for mobile devices).

The display technology used for visualization - the XR device - will be based on true 3D Holography, an interference-based technology (cf., interference of light) where the user can see depth in space (true 3D) and can focus the details of the 3D-avatar or the 3D-information shown in space with his eyes (like with camera changing focus). Overall, this type of display provides the same natural viewing experience for the user's eyes as if the holographic assistant would be a real object or person. The additional information visualized could be text data (written information), image data (photos, maps etc.) or 3D-data (3D-models, 3D-visualizations).

4.2.3.1 Application Development

During this phase, SRT implements a new Holographic Assistant (HA) application, directly following the given structure, the CHARITY XR platform offers. The CHARITY XR platform is aligned with the requirements of the use case, such as supporting high end GPU powered Windows Server based VMs (preferably on edge), able to run the Unity 3D platform for real-time rendering and also supporting GPU accelerated CUDA code, also preferably on edge.

SRT is proposing a visual system where a three-dimensional (3D) avatar – assistant – appears on a holographic 3D display to offer assistance, such as answering questions or showing information compiled from raw data sources (see Figure 46). Specifically, the cases: showing weather information, showing information about shares from stock exchanges and a chatting feature. To gather the required information, raw data is requested from Internet Service Providers.

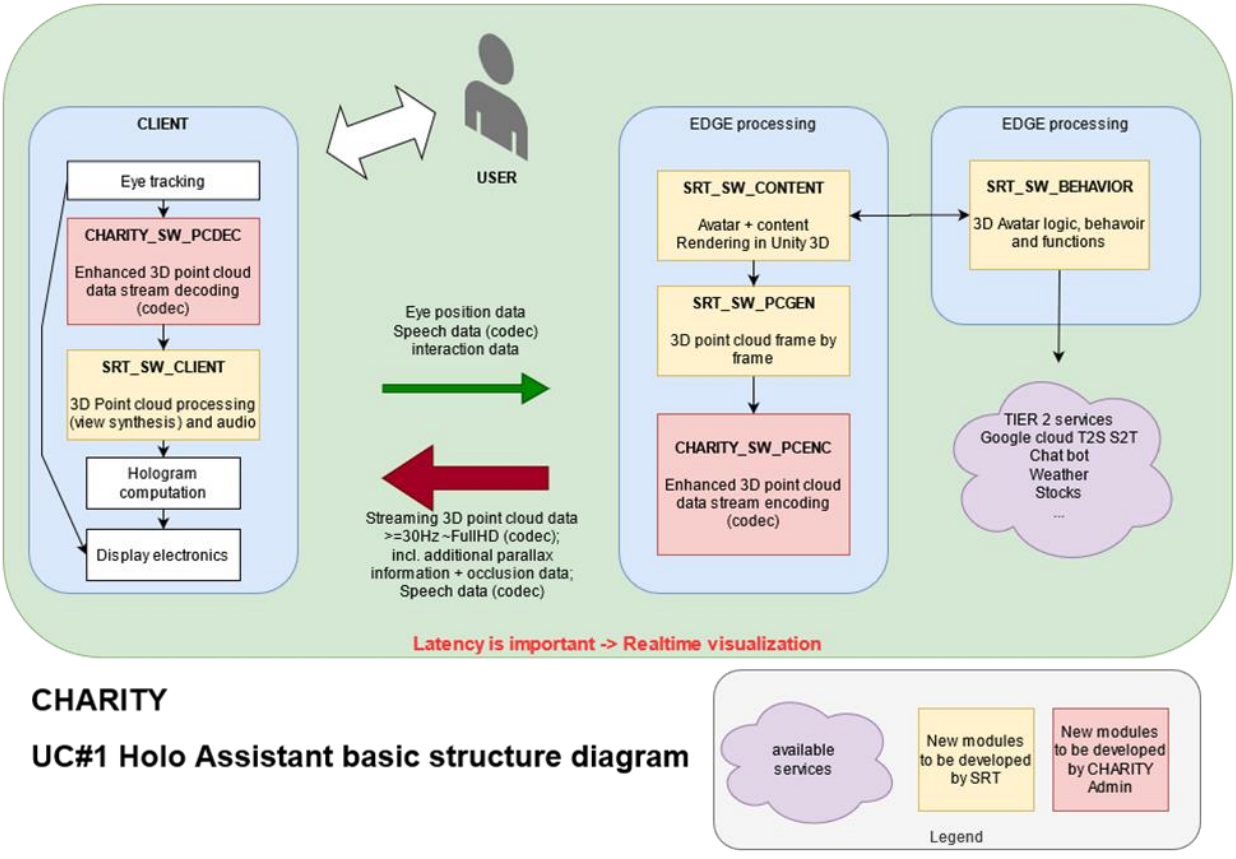

Figure 46: Basic overview of the components of SRT APP and how they interact

The SRT Application **Holographic Assistant** (HA) or SRT APP mainly consists of following components running on the CHARITY XR platform:

- **SRT_SW_CONTENT**, VNF, developed by SRT
  - Responsible for compositing the 3D content showing the animated avatar of the HA and compiled information (uses Unity 3D for rendering)
  - Rendering the dynamically allocated amount of views required for point cloud (PC) generation
  - Provides actuators: ***RANGE*** and ***STEPWIDTH*** to tune number of views to be generated for feeding into **SRT_SW_PCGEN**
- **SRT_SW_PCGEN**, VNF, developed by SRT
  - Responsible for generating a point cloud data set (PCDS) from the number of views provided by **SRT_SW_CONTENT**
- **CHARITY_SW_PCENC**, VNF, developed by CHARITY Admin
  - Responsible for compressing the PCDS and optionally making use of PCDS from last or even former frames to transfer only differential data
  - Transfers compressed PCDS to **CHARITY_SW_PCDEC** running on SRT Client
  - Provides actuators: ***COMPRESSION_RATIO*** to tune compression ratio of PCDS transferred over the network / WAN, the ratio has the goal to provide best QoE for End user possible at current network conditions, a higher ***COMPRESSION_RATIO*** automatically means a certain reduction in visual quality, but with lower ***COMPRESSION_RATIO*** the quality change may not be sensed by the End user.
- **CHARITY_SW_PCDEC**, developed by CHARITY Admin
  - Responsible for decompressing the received PCDS (compressed) and providing the current PCDS to **SRT_SW_CLIENT** for processing and visualization on the 3D holographic display
  - Provides monitors: ***LATENCY***, ***FRAMERATE***, and ***DATARATE*** to provide current network status to Analytics Engines and appropriately control performance of the components via Decision Engines and Actuation Engines controlling the actuators listed above
- **SRT_SW_BEHAVIOR**, VNF, developed by SRT
  - Responsible for processing input from End user (speech recording from **SRT_SW_CLIENT**), gathering and providing information for visual composition by **SRT_SW_CONTENT**, providing speech output back to End user (answer speech data to **SRT_SW_CLIENT**)
- **SRT_SW_CLIENT**, developed by SRT
  - Responsible for bringing the PCDS data to the holographic 3D display, capturing End user speech input, providing speech output provided by **SRT_SW_BEHAVIOR**

The monitors listed above are evaluated in the Domain Specific XR Services Monitoring & Reaction Plane to drive the appropriate actuators also listed above. The exact behaviour will be implemented in appropriate Analysis, Actuation, and Decision Engines and will be tuned within empiric tests at the running system. So the performance of the whole system can be adapted in real time with the goal of achieving best possible QoE for the End user. The Analysis, Decision, Actuation engines could be some standard engines provided by CHARITY Admin which are configured with allowed parameter ranges and appropriate parameters to change when these limitations are stressed. No XR specific logic is needed, a simple parameter based IFTTT (if this then that) approach would be sufficient.

The VNFs SRT_SW_CONTENT, SRT_SW_PCGEN, CHARITY_SW_PCENC share the same GPU memory for data exchange, due to heavy bandwidth requirements of 100+ Gbit/s and low latency requirements. This is given by the requirement to render and process a certain number views (Full HD-resolution) from the same Unity 3D application scene into a PCDS, with overall frame-rate of 30Hz minimum. Overall target is to reduce E2E latency between SRT_SW_CONTENT and CHARITY_SW_PCDEC to a minimum of < 200 ms.

Figure 47 shows the data flow and relation between the SRT HA specific VNFs / local components running on the CHARITY XR platform. All VNFs are implemented as virtual machines (VMs) and preferably run in the edge in the same domain. The local components providing monitors (via Monitoring agents and XR data collectors) and VNFs providing actuators (driven via XR specific Actuation Engines) are located in different domains (local + edge), but the managing components of

Domain Specific XR Services Monitoring & Reaction Plane specifically for SRT HA run in the same domain as the VNFs, esp. the Analysis, Actuation, and Decision Engines and XR data collectors. So, there is no CHARITY XR platform cross domain interaction needed. Interactions / communication will only be within same domain or to the outside to local system or internet.

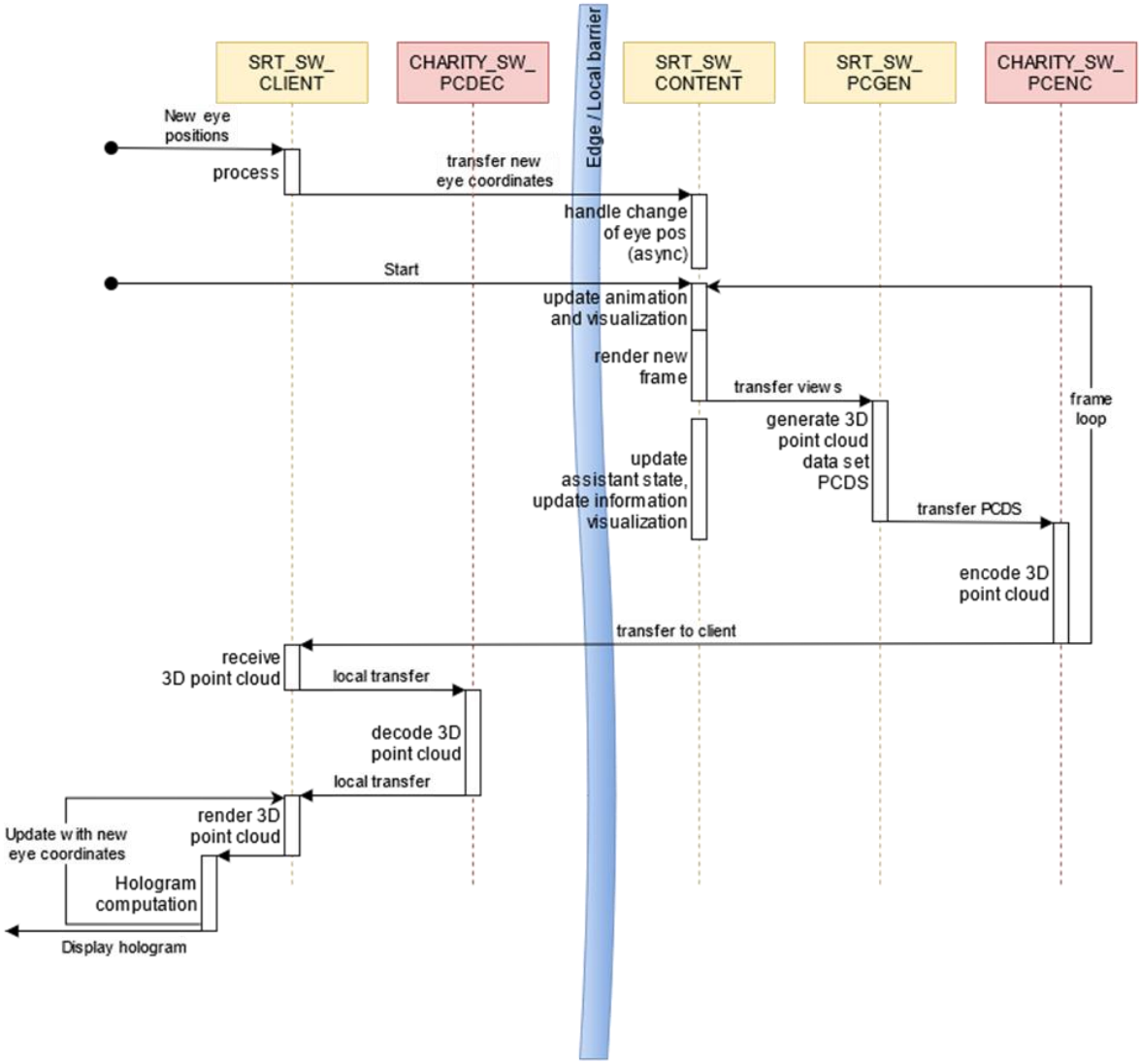

Figure 47: Relation of all VNFs and local components for the SRT HA application use case

At the end of this phase, all VNFs/VMs needed for SRTs HA will be uploaded to 'XR Service Enabler Repository'. SRT will also provide information how all the components relate to each other, which of them are connected how and which components need to be instantiated to properly enable the XR service. This resulting "blueprint", containing the description of all necessary functional & non-functional requirements as well as the topology of different components, will be uploaded to the 'XR Service Blueprint Template Repository'.

4.2.3.2 Application Deployment

In this phase, SRT will deploy SRT APP on the CHARITY platform. This will strictly follow the deployment concept already explained in section 3.4. This consists of selecting the blueprint mentioned above from the 'XR Service Blueprint Template Repository' and sending it to CHARITY XR platform for deployment. Once CHARITY receives the blueprint, it will use the 'XR Service Enabler Repository', 'Resource Planning', 'Resource Indexing', and 'Plane Services Registry & Discovery' entities in order to schedule the XR service. According to the blueprint information and data received from the SRT Client, CHARITY will search and select the optimal domain that the VNFs of the SRT HA application will be deployed.

Each deployed component instance set of all VNFs will be assigned to one SRT Client. At this stage, some components need to be ran in the edge while the others run locally on SRT Client.

The CHARITY orchestrator will deploy the components at the selected optimal domain. Upon creation, the different components will register themselves in the 'Plane Services Registry & Discovery'. The registered components will use this plane to connect to each other. For instance, all monitoring agents will use that mechanism to find the XR data collector service to which they will be sending the monitored data.

Once all components are up and running, the End user can begin using the SRT HA application - the SRT Client already connected to the software components in edge. Now also the blueprint of the XR service was added to 'Running XR Services Repository'.

4.2.3.3 Application Exploitation

At this phase, the HA application would be running, while the End user would interact with the virtual assistant shown on the Holographic 3D display within the SRT Client. During this phase, CHARITY XR platform would be responsible for ensuring the satisfaction of the KPIs, meaning that all closed loops would be functional. In what follows, we will consider some hypothetical scenarios to show the interplay between the different components.

- The ***FRAMERATE*** of received 3D point cloud stream falls below given limit for a certain amount of time. Monitoring agents in **CHARITY_SW_PCDEC** will continuously provide data to XR data collectors in the XR service monitoring and reaction plane about current frame rate. After analysis in Analytics Engine the XR service Decision Engine will immediately act and will choose and adapt via Actuation Engine suitable parameters for compression of the 3D point cloud data. The situation will be normalized again if network conditions relax.
- The E2E ***LATENCY*** of received 3D point cloud stream falls below given limit for a certain amount of time. Monitoring agents in **CHARITY_SW_PCDEC** will continuously provide data to XR data collectors in the XR service monitoring and reaction plane about current E2E latency. After analysis in Analytics Engine the XR service Decision Engine will immediately act and will choose and adapt via Actuation Engine suitable parameters for number of encoded views of the 3D point cloud data to increase the range from which the data is correctly visible (actuators ***RANGE*** and ***STEPWIDTH***). The situation will be normalized again if network conditions relax.
- The ***DATARATE*** of received 3D point cloud stream is below the limit to achieve a certain frame rate for a certain amount of time. Monitoring agents in **CHARITY_SW_PCDEC** will continuously provide data to XR data collectors in the XR service monitoring and reaction plane about current data rate available through the network. After analysis in Analytics Engine the XR service Decision Engine will immediately act and will choose and adapt via Actuation Engine suitable parameters for compression of the 3D point cloud data (actuator ***COMPRESSION_RATIO***). Also, the number of views could be reduced in the PCDS and possibly the step width is increased (actuators ***RANGE*** and ***STEPWIDTH***). This may lead to reduced quality but will maintain at least a minimum frame rate under such conditions. The situation will be normalized again if network conditions relax.
- Communication with a certain component immediately stopped. The charity platform will detect this since monitoring of the component also failed. The CHARITY XR platform will be responsible to create new instance of the missing component possibly on a different location. Since the VNFs are stateless, at least from session to session, the instances can be recreated without dependency from the crashed instances. If the **SRT_SW_BEHAVIOR** VNF needs to be recreated, the current state of the SRT HA application may be lost, but in this case the application may ask the End user to repeat his request, which is acceptable.
- The E2E Service Specific AE predicts that the edge domain containing the XR-APP component would be overloaded soon. So the E2E Service Specific DE decided to perform a service migration. Given the VM nature of this application, the DE decided to deploy a new set of components in a new Edge domain. The handover will happen between two sessions (i.e.,

between two requests). Once the handover is completed, the components of the old edge domain are terminated.

Obviously, it is the responsibility of the Actuation Engines to carry out and enforce the decisions made by the different Decision Engines. Also, it is worth mentioning that all taken actions are validated by ensuring that they fall within the resource range available for SRTs HA.

4.2.3.4 Application Decommissioning

Finally, during this phase, the resources are reclaimed. The termination actions provided by SRTs HA are executed first; they are part of the XR service blueprint. The blueprint inside the 'Running XR Services Repository' is marked as 'terminating' and it is deleted once all the related components are decommissioned. Once the HA application is terminated, the related shared services are inspected to be also decommissioned. If the shared services are not being used by other XR services, they are terminated. Otherwise, they will be kept, but the closed loops inside them will eventually reduce the amount of committed resources to those services. The VNFs are basically stateless, thus it is not required to save the state of the software running.

4.2.4 UC2.1 – VR Medical Training

UC2-1 regards an immersive collaborative VR medical training application. It is based on ORAMA's software platform, MAGES SDK, which is a multi-player game engine working on top of standard networking protocols. Achieving QoE in such a multi-player environment is a challenging subject with low latency being a critical factor. All training scenes, moving objects as well as transformations and deformations of 3D soft-body objects require a low latency - high bandwidth network as well as a significant amount of GPU and CPU processing power, for mathematical calculations, and rendering. MAGES SDK natively supports multiple CCUs as its networking functionality operates beyond the standard client-server model, using a relay server for transferring the current state of deformable objects.

4.2.4.1 Application Development

To exploit the various data services, low latency and the advanced processing capabilities that CHARITY framework provides, through available resources across the cloud-edge compute continuum, the ORAMA software platform is scaling to edge/cloud resources. This will be accomplished by offloading heavy computational components to nearby edge resources. Currently, the adapted UC workflow adopts the micro-service concept as the main application will be deployed as virtual machine (VM) stateful micro-service on edge. One instance of the application will be deployed for each HMD. The application instance will be responsible of computing, rendering, and encoding the images that will be transmitted to the HMD by a signalling server. Also, based on the received network characteristics, the application-instance will provide run-time adaptation and dynamic optimization of the Geometric Algebra Interpolation Engine. The lightweight HMDs will be responsible for decoding and projecting the transferred images from the edge, and to capture and transfer user events (controllers' position, triggers, etc.) to the application instance. Also, current research activities focus on altering the UC workflow by dissecting the physics engine from the main application micro-service to a separate VM micro-service that will be deployed in a nearby edge.

During the application development phase, ORAMA implements its XR service. The ORAMA team begins by developing the different components needed for their service while following the micro-service paradigm. This consists in dividing the application logic into different components that can be ran independently. Communications between these components are achieved via network sockets. ORAMA has divided its application logic into four components, namely Signalling Service, LSpart_1, LSpart2, and Relay Server. The Signalling Service component will be responsible of transmitting images to the HMDs using WebRTC. The LSpart_1 component will be responsible of maintaining the application's main logic and of rendering procedures. The LSpart_2 will contain the Physics engine performing soft-body deformation calculations. The Relay Server, that employs the Photon Relay

Server, is responsible of maintaining the current state (scene graph) of each virtual operating room and of transmitting it to all end-users.

Having already a functional application, ORAMA will deploy it on the CHARITY platform. In order to do so, an integration phase is needed. It consists in defining the desired QoS, the metrics and the monitoring agents needed to assess the performance of the application. Among others, the QoS should encompass at least information about the E2E latency requirements and also an indication about the expected bandwidth requirements between the components. ORAMA needs to define how its application should adapt to avoid performance degradations, by defining/extending/altering specifically tailored algorithms for the MS, AE, and DE. This can be done by re-using existing elements in 'XR Service Enabler Repository', by mixing some elements from there, or even implementing and publishing new elements. For instance, ORAMA may connect its three components with a monitoring server that gathers the information of resource and network usage.

At the end of this phase, all artefacts needed for ORAMA-VR will be uploaded to 'XR Service Enabler Repository'. ORAMA will also define a blueprint that describes how all the components relate to each other. This resulting blueprint, containing the description of all necessary functional & non-functional components as well as the topology of different components, will be uploaded to the 'XR Service Blueprint Template Repository'.

Below you may find some indicative requirements and how they are addressed by the CHARITY architecture components.

- The HMD app should be able to connect via standardized protocols to the deployed resources (LS_PART1 and LS_PART2 instances on edge) where part of the application has been offloaded: The *XR Service E2E conducting plane*, as the entry point for XR providers, may address this requirement, by utilizing the *XR service exposure* and the *Running XR Services Repository* components.
- The resource discovery mechanism of CHARITY should efficiently deploy to nearby edge resources considering lowest average latency: The *XR Service E2E conducting plane*, through its *Plane services Registry & discovery component*, locates the appropriate nearby edge resource. The *Service planning and deployment*, the *XR Service Blueprint Templates Repository*, and the *Running XR Services Repository* components are utilized for the planning the deployment process. Furthermore, the XR *Service Deployment Plane* is also involved for the deployment and hosting of the LS_PART1 VM (Geometric Algebra Interpolation Engine) and LS_PART2 VM (Physics Engine).
- Different instances of the application can run on the same edge node supporting different users via their HMD devices: This requirement is addressed by the *Service planning and deployment* component of the *XR Service E2E conducting plane*.
- Continue using the application in case of problems in network resources with minimal delay: The *Domain specific XR Service Monitoring & reaction Plane*, through its *E2E analytics engine* and *E2E decision engine* components, in cooperation with the *Monitoring agents*, are responsible of detecting potential network resource problems and decide to instruct the XR service to adapt accordingly the produced resolution in order to achieve QoE.
- Receive error messages on potential problems with existing resources, continue the VR app by communicating with another newly discovered resource (discovery and placement): The *Domain specific XR Service Monitoring & reaction Plane*, through its *E2E analytics engine* and *E2E decision engine* components, in cooperation with the *Monitoring agents*, are responsible of detecting potential resource problems and decide the discovery of new nearby edge resource. The deployment of the new micro-services, replacing the original ones, is accomplished utilizing similar process to the main XR service deployment.
- Able to visualize performance analytics on the HMD: This information is generated by the system analytics component of the XR service in cooperation with the *E2E analytics engine*,

and the E*2E decision engine* components of the *Domain specific XR Service Monitoring & reaction Plane*, and the *Monitoring agents*.

4.2.4.2 Application Deployment

In this phase, ORAMA will deploy its application (ORAMA-VR) on the CHARITY platform (Figure 48). This consists into selecting a blueprint from the 'XR Service Blueprint Template Repository' and customize it before sending it to CHARITY for deployment. Once CHARITY receives the blueprint, it will use the 'XR Service Enabler Repository', 'Resource Planning', 'Resource Indexing', and 'Plane Services Registry & Discovery' entities in order to schedule the XR service. According to the information of the end users and the chosen enablers, CHARITY will search and select the optimal domains where the different components of the XR service will be deployed. Each deployed component instance will be assigned to a single HMD. At this stage, all components, Signalling Service, LSpart_1, and LSpart2, need to be ran in the edge while the Relay Server and the monitoring server would be deployed in the cloud.

The CHARITY orchestrator will deploy the components at the selected optimal domains and stitch these domains together to offer seamless connectivity between the components. Upon creation, the different components will register themselves in the 'Plane Services Registry & Discovery'. The registered components will use this plane to connect to each other. For instance, all monitoring agents of Signalling Service, LSpart_1, LSpart2, and Relay Server will use that mechanism to find the monitoring server to which they will be sending the monitored data.

Once all components are up and running, the end users can begin using the XR service. Indeed, the end-users' devices can connect to the Signalling Service and the LSpart_1, which gathers more processed data from LSpart_2. It is worth mentioning that by this time, the blueprint of the XR service will already be added to 'Running XR Services Repository'.

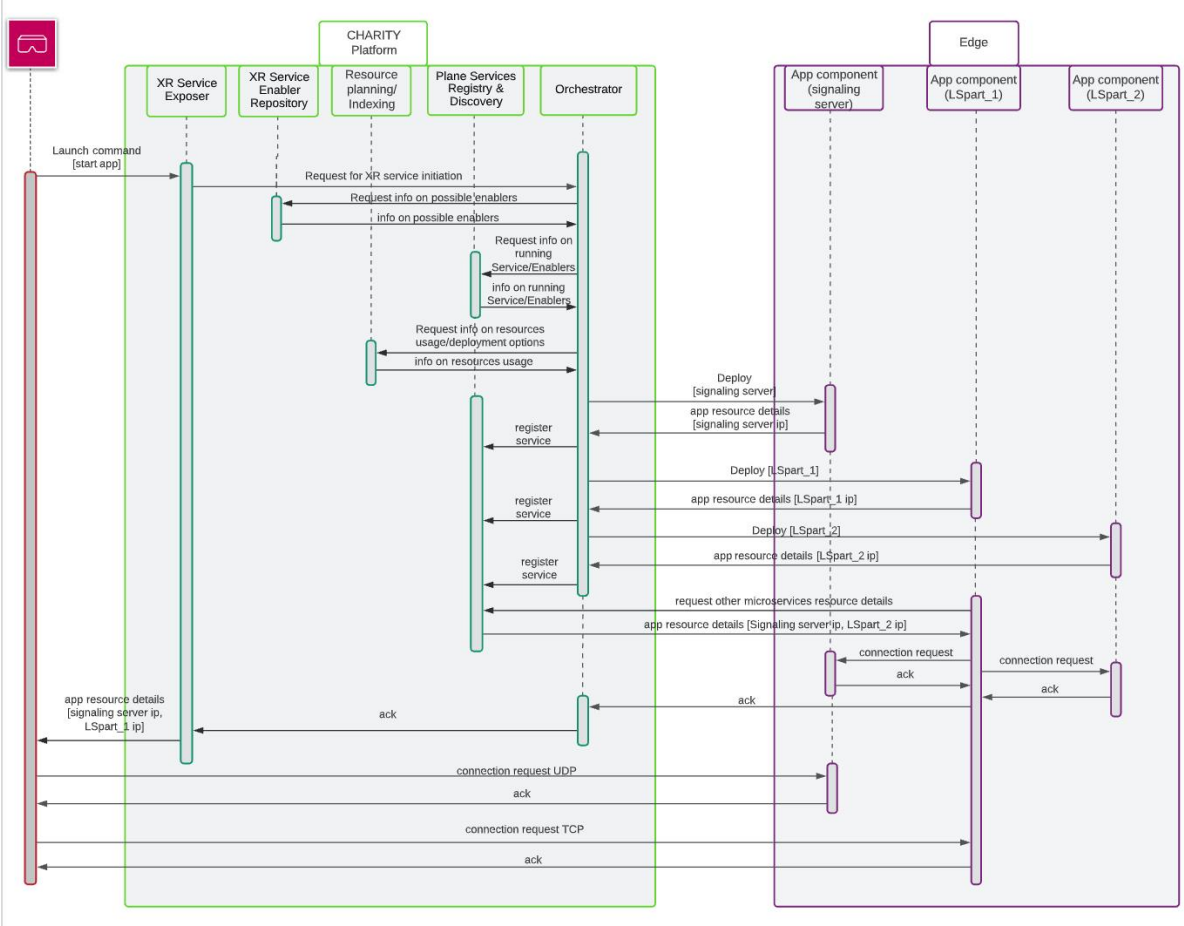

Figure 48: Application deployment sequence diagram

4.2.4.3 Application Exploitation

At this phase, ORAMA-VR would be running, while the end-users would be performing a VR medical training. During this phase, CHARITY would be responsible for ensuring the satisfaction of the KPIs, meaning that all closed loops would be functional. In what follows, we will consider some hypothetical scenarios to show the interplay between the different components.

- Let's suppose that a VR medical training is underway and a new end-user wants to collaborate in the same surgical room. Given the location of the new user, CHARITY will configure and deploy app components (e.g., LSpart1) in an edge cloud close to the new end-user. Once the app components are up and running, they register themselves in the 'Plane Services Registry & Discovery' and they also use it to find how to connect to the relay server and the monitoring server. The new end-user's ORAMA-VR instance will query Photon relay server for available surgical rooms and select the appropriate one. The needed changes will be forwarded to 'Running XR Services Repository'.
- After a short period of time, the Relay Server became overloaded (e.g., due to the increase of the number of users). The resources usages are gathered by the MS and are used by the local AE that will predict that this overload can result in a degradation of the QoS and thus will alert the local DE of this overload. The local DE will allocate more resources for the Relay Server which will solve the issue for a time.
- After a period of time, the E2E analytical engine predicts that the latency between the different domains would increase (e.g., due to time of the day) which would degrade the QoS of some of the users. Once the E2E DE is alerted, it will try to redistribute the latency budget between the different domains (e.g., increase resources usage to reduce processing time). But unfortunately, most of the local DE, which are edge domains, would not be able to satisfy the new latencies' budgets. After that, the DE will decide to migrate the Relay Server to a central position that minimizes the latency between the Relay Server and the other components that are deployed at the edge.
- As time goes on, the congestion of the network increases which will threaten QoS of ORAMA-VR application. Failing to find a solution by increasing resources or by migrating components, the E2E DE will decide to decrease the quality of the VR experience to maintain the usability of the application. These algorithms are application specific, so they are the components defined by ORAMA during the first phase (i.e., Application Development in section 4.2.4.1). These can consist in reducing the frame rate, the resolution, etc.

Obviously, it is the responsibility of the Actuation Engines to carry out and enforce the decisions made by the different DEs. Also, it is worth mentioning that all taken actions are validated by ensuring that they fall within the resource range available for ORAMA.

4.2.4.4 Application Decommissioning

Finally, during this phase, the resources are reclaimed. The termination actions provided by ORAMA are executed first; they are part of the XR service blueprint. The blueprint inside the 'Running XR Services Repository' is marked as 'terminating' and it is deleted once all the related components are decommissioned. Once the ORAMA-VR application is terminated, the related shared services are inspected to be also decommissioned. If the shared services are not being used by other XR services, they are terminated. Otherwise, they will be kept, but the closed loops inside them will eventually reduce the amount of committed resources to those services.

4.2.5 UC2.2 – VR Tour Creator

The VR Tour Creator, use case of CHARITY, is a platform that allows the user to create VR tours very easily. This platform can be used for learning, storytelling, marketing, real estate and lots of other businesses. This platform consists of several modules on the back-end and on the front-end. The front-

end modules consist of the back-office and the viewer; the back-office, referred as Cyango Cloud Studio, is a web-app that manages the tours created by the user; and the viewer is referred as Cyango Story, which enables the user to view and consume the content created by the Cyango Cloud Studio. The back-end consists of an API containerized in Docker. All media are stored on Amazon Web Services, while the VR video is converted to proper format using Amazon Media Converter. CHARITY architecture will make the video conversion and the streaming delivery faster.

4.2.5.1 Application Development

In this phase, DOTESFERA will be developing and implementing multiple components to guarantee the function of the platform using the micro-service paradigm. All these components will be running under Docker environment and will communicate with each other. Most of the components will communicate with the API component which manages the other components to use.

The 2 front-end applications (Cyango Story and Cyango Cloud Editor), which can be accessed by the end-user, are built using the React.js Framework which after compiled can be served on a simple NGINX server. Each application is served on a Docker container which contains many running services like Node.js and Nginx to properly serve the application.

The API will also be running under a Docker container with the necessary services like Node.js and Nginx.

The database will be running on a separate Docker container using the official MongoDB Docker image with all the needed services to work.

Other components like the Media conversion will be developed and tested using open-source applications like FFMPEG. Right now we are using Amazon Web Services Media Converter to convert the uploaded videos by the user to the HLS standard format and host it on AWS S3 ready to be consumed by the front-end applications. To migrate to CHARITY cloud it is needed to build a component to serve the same purpose as AWS Media Converter, and this converter component will be containerized and running under CHARITY cloud.

After all these developments the next phase is the integration phase of all the components in the CHARITY cloud that require monitoring and metrics evaluation to improve the quality of the system and reach the bandwidth and latency estimated throughout all the components.

Dotesfera will define the blueprint that describes how every functional and non-functional component works and communicates with others. It will also populate the ‘XR Service Enabler Repository’ with built artefacts.

4.2.5.2 Application Deployment

In this phase will the CHARITY orchestrator will deploy the components and assign each one a different alias for the components to easily communicate with each other. CHARITY will assign the nearest cloud cluster depending on the user’s location and information to provide the best quality of experience. The user can be located anywhere in the world, and the quality of the experience needs to be independent of the user’s location. The Application Provider would send a blueprint to the orchestrator detailing the user’s location and the relevant XR enablers needed for the XR application. The application deployment will consist on deploying all the docker containers built with all the applications and scripts needed to run.

4.2.5.3 Application Exploitation

At this phase, Cyango would be running on the CHARITY, and the end-users will be using the platform proceeding to a variety of features they can use. CHARITY will be responsible to ensure the achievement of the KPI’s.

Here are some possible scenarios of the interaction between each component:

- A user uploads a heavy VR video. CHARITY will configure and deploy the Media convert component in the closest cloud edge to the user. Then this component will start processing, converting and optimizing the video, then it will host the video on the cloud with a unique address to be stored on the database with the relative data.
- Many users close to each other would begin requesting the same Cyango Story. CHARITY will enable pro-active caching and will cache the corresponding data close to the users while following the KPI's metrics.
- The user starts editing a video on the Cyango Cloud Studio. Editing the video requires a very fast communication between the user and the cloud. Every time a user applies some video editing action (e.g., trim, colour correction, etc.), they are ran on the cloud to spare the user's device processing and heavy data management. Therefore, if the link between Cyango Cloud Studio and the user deteriorates (e.g., due to user mobility), CHARITY would perform a migration of the component to a location close to the user. Given the fact that this component is stateful, the timing for the migration would be chosen in concertation with the XR application.

Also, it is worth mentioning that all actions taken are validated by ensuring that they fall within the resource range available for Dotesfera.

4.2.5.4 Application Decommissioning

Finally, during this phase, the resources are terminated. The termination actions provided by Cyango are executed first. Once the main components are terminated, the related shared services are inspected to be also terminated. If the shared services are not being used by other XR services, they are terminated.

4.2.6 UC3.1 – Collaborative Gaming

UC3.1 will use CHARITY resources, network orchestration capabilities and XR Services. XR Resource Discovery, Indexing and Planning will be used for optimal resource allocation to enable the Game Server automatic deployment process. XR Resource Monitoring, XR Services Blueprint Templates and Enablers will be used for deployment maintenance, automatic Game Server deactivation and resource recovery.

4.2.6.1 Application Development

During this phase, ORBK will be developing three components that make up AR Collaborative Gaming application: **Game Client**, **Game Server** and **Game Servers Status DB**. Each component of the UC3.1 will be responsible for handling one of the three key functionality of the UC 3.1 system:

- **Game Client** will be an iOS application running on Apple devices. Game Client will be responsible for presenting the current state of the game virtual environment mixed with the camera feed of the surrounding environment. It will also gather information about the user's current position and rotation in order to synchronize it with other users through the Game Server. Game Client will also allow you to interact with the application gathering and interpreting the user's input. Due to the requirement to use LiDAR the device must be equipped with this technology in order to be able to efficiently scan and interpret environment data.
- **Game Server** will be a central point for each game session. It will be a program compiled to run on Linux platforms. The role of the Game Server is to gather all game information from all Game Clients, perform game simulation and to send the state of all game objects back to all

connected clients. Direct communication between Game Clients and Game Server will be realized using the UDP protocol.

- **Game Servers Status DB** will be a central service for all deployed Game Servers responsible for providing information about deployed Game Servers to the Game Clients. It will also provide information about deployed Game Servers to the CHARITY platform.

UC 3.1 uses services provided by CHARITY platform:

XR Resource Discovery, Indexing and Planning

- optimal resource allocation for Game Server deployment
- automatic and efficient deployment process

XR Resource Monitoring

- automatic Game Server deactivation mechanism and resource recovery
- XR Services Blueprint Templates and Enablers for deployment maintenance

Specialized XR Services

- rapid mixed reality data synchronization

To achieve relevant KPIs and maintain QoE, several monitoring functions and control mechanisms will be implemented at the various modules to react to changing network conditions or requirements.

At the end of this phase, all components needed for ORBK AR Collaborative Gaming will be uploaded to XR Service Enabler Repository. ORBK will also define a blueprint that describes how all the components relate to each other. This resulting blueprint, containing the description of all necessary functional & non-functional components as well as the topology of different components, will be uploaded to the XR Service Blueprint Template Repository.

4.2.6.2 Application Deployment

In this phase, ORBK will deploy Game Servers on the CHARITY platform (Figure 49).

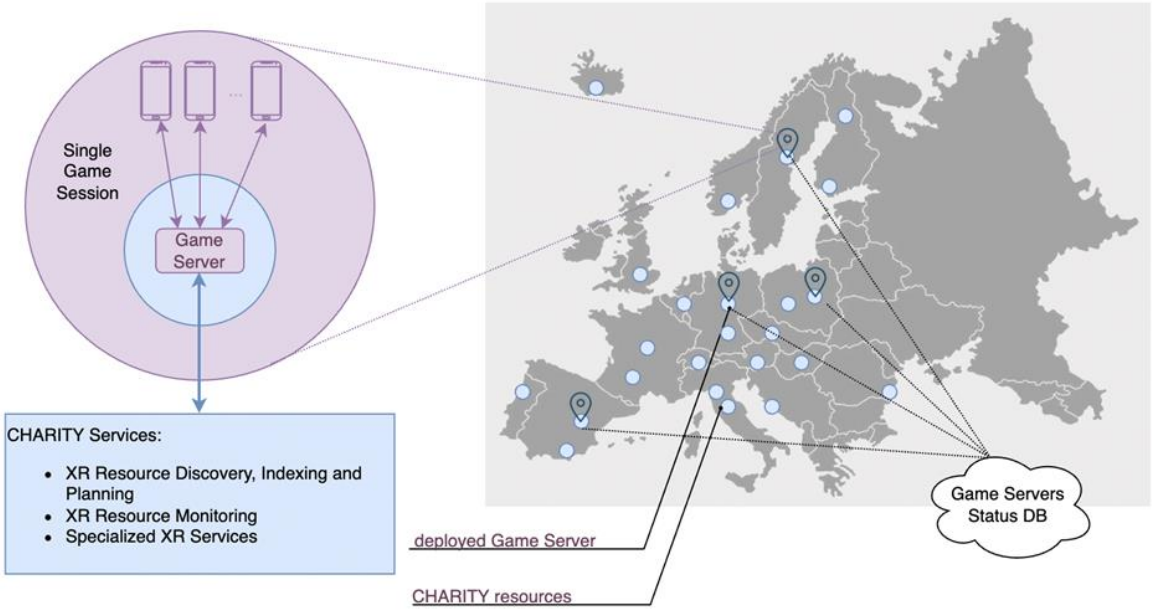

Figure 49: High overview of ORBK AR Collaborative Gaming deployment on CHARITY platform

For each game session, a Game Server must be deployed. A single blueprint from the XR Service Blueprint Template Repository will be defined in the Application Development phase for the Game Server deployment. CHARITY Platform will be able to use it for the automatic deployment of each

Game Server. The blueprint will contain a description of the relation between UC components, expected bandwidth and latency requirements between the components. CHARITY Platform must use the XR Service Enabler Repository, Resource Planning, Resource Indexing, and Plane Services Registry & Discovery entities in order to deploy the UC 3.1 application.

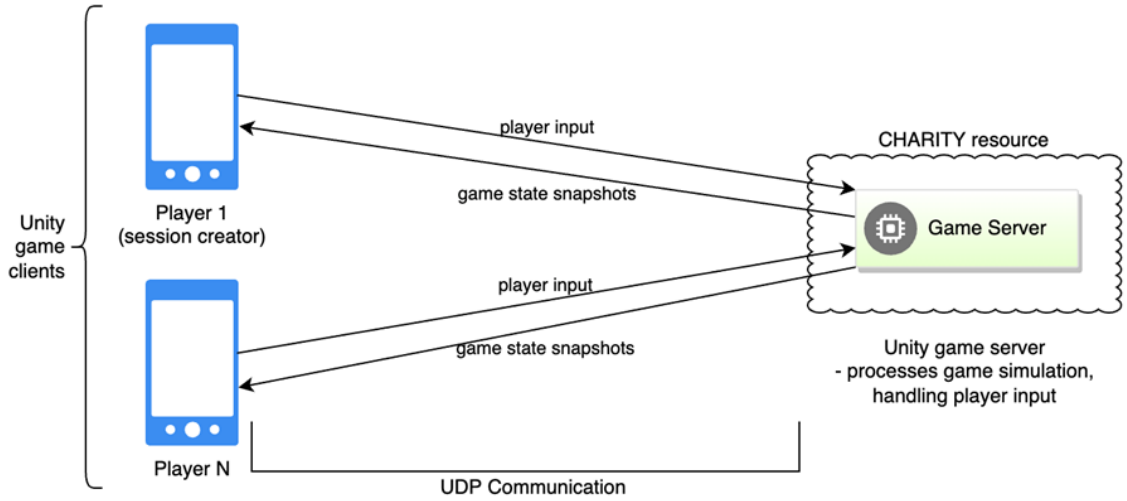

Figure 50: Game Server proceeding

Game Servers will be deployed on the resources provided by CHARITY platform. CHARITY must be able to provide optimal resources for a given group of players taking into account their location and network connection latency from server to clients.

According to the information of the Game Clients and the chosen enablers, CHARITY will search and select the optimal edge resources for the deployment of Game Server components. ORBK will provide Game Server in the form of Docker Image.

Game Servers Status DB will be deployed outside of CHARITY Platform.

Game Client will be outside of the CHARITY Platform, available as an installation file for download and install on iOS devices.

Application Game Server component provisioning

The UC 3.1 provider can upload a Game Server docker image to the 'XR Service Enabler Repository' using CHARITY Platform 'XR Service Exposure' along with the Game Server deployment blueprint (Figure 51).

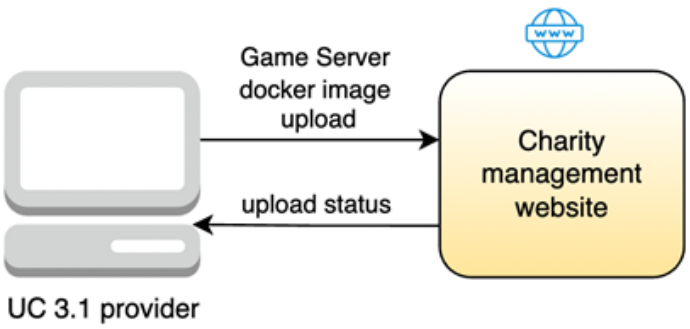

Figure 51: Upload application artefacts and blueprint

Single Game Session deployment

In order to run a single Game Session, the Game Servers Status DB would reach out to CHARITY to deploy a Game Server instance (Figure 52 and Figure 53). Considering the location of the users and their number, CHARITY would leverage 'Resource Indexing', 'Resource Planning' in order to deploy the Game Server in the right Cloud or Edge location. CHARITY would take into consideration the fact that the Game Server is able to be directly reachable by Game Clients using UDP protocol. Once the Game Server is deployed, CHARITY would send its IP address to Game Servers Status DB.

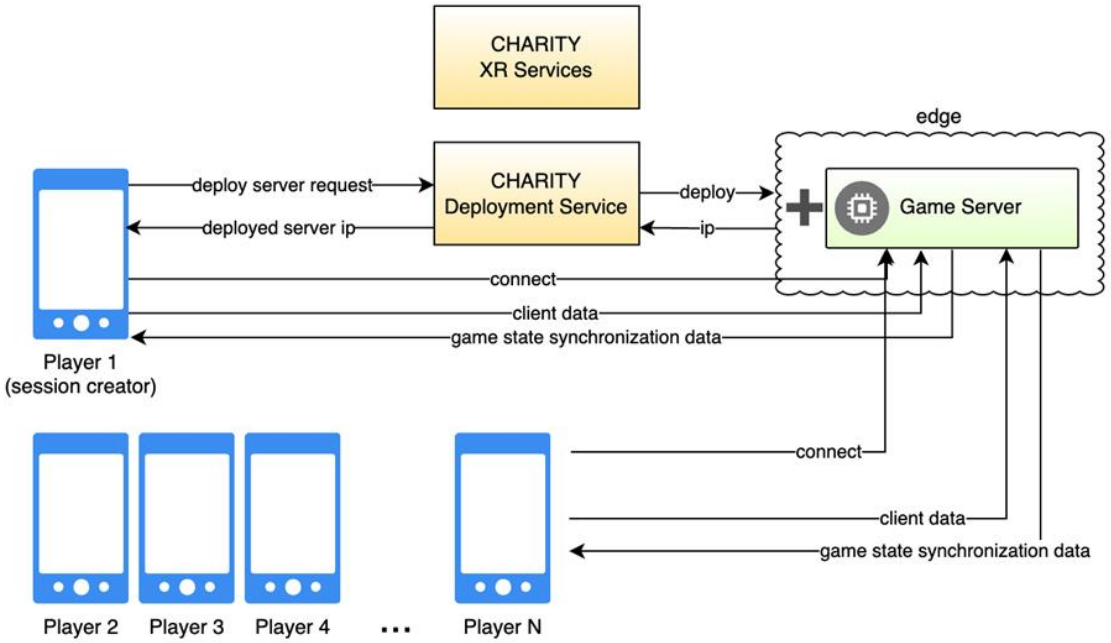

Figure 52: Game Session architecture

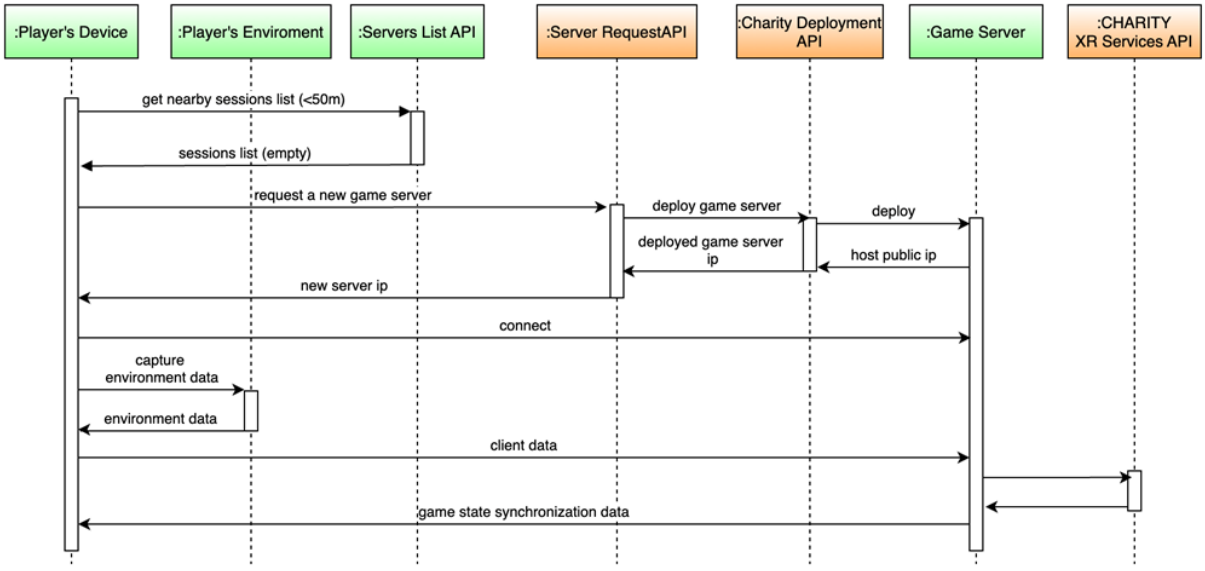

Figure 53: Deployment of Game Server and start of Game Session

Multiple Game Sessions deployment

CHARITY Platform must be able to deploy multiple Game Servers. The resource that will be used for the server deployment must be optimally selected by the CHARITY orchestrator using XR Resource Discovery, Indexing and Planning services to provide minimal network latency and bandwidth.

4.2.6.3 Application Exploitation

AR Collaborative Gaming application will be able to run and players (end-users) will be able to interact with each other in the extended reality provided by the application. Players will be able to see through their mobile devices both real environments, with use of the cameras and augmented objects that will be added to create an impression of extended reality. Players will be sharing the same space, both real and virtual. All augmented virtual objects will be synchronized between all players in the same game session. CHARITY Platform will monitor the resource load and network latency between clients and servers in order to provide end users with satisfactory QoE.

Possible scenario of CHARITY operation to ensure high QoE:

- Multiple Game Sessions are running in different geolocations and multiple Players are connected to each game session.
- Some time later, in one of the Game Sessions a drop in QoE is detected by CHARITY monitoring services. The drop is possible if many clients join a single Game Session and the network resources are no longer able to handle Game simulation processing (because of large number of in-game objects) or the network connection causes greater than the allowable increase in latency.
- CHARITY Platform tries to find and allocate more efficient network resources for the given Game Session. If the previous operation is successful new Game Server is deployed and Players connected to the given Game Session are informed. Game Session can be restarted using new Game Server.

4.2.6.4 Application Decommissioning

Finally, during this phase, the resources are reclaimed.

When all the players disconnect from the Game Server the Game Session must be terminated. All connections with Clients must be closed and the Game Server must be removed from the resource provided by CHARITY.

The termination actions provided by ORBK are executed first; they are part of the XR service blueprint. The blueprint inside the Running XR Services Repository is marked as terminating and it is deleted once all the related components are decommissioned. Once the AR Collaborative Gaming application is terminated, the related shared services are inspected to be also decommissioned. If the shared services are not being used by other XR services, they are terminated. Otherwise, they will be kept, but the closed loops inside them will eventually reduce the amount of committed resources to those services.

4.2.7 UC3.2 – Manned-Unmanned Operations Trainer

This use case enables a collaborative immersive training environment of emerging civil manned-unmanned teaming concepts while minimizing the involvement of expensive equipment. The key goal is to advance the deployment of training simulators on the cloud-edge continuum and target XR devices to deliver compelling immersive environments with minimal local technology assets. The fluidity of CHARITY cloud/edge resources and network orchestration is leveraged to facilitate engagement and collaboration between multiple simulation instances.

4.2.7.1 Application Development

In the development phase, various necessary XR services for Collins' flight simulation use case (Collins-XR) are implemented. The implementation follows the micro-services paradigm and takes advances in containerization for application's agility and mobility. Collins looks to cloud/edge computing to engineer a new way forward that enables a high-end collaborative training simulation environment in

Collins-XR. In this way, Collins has distributed its application logic over three layers, namely: device (i.e., contains Flight Dynamics, Interaction Management, and View Management), edge (i.e., contains Flight Oracle and Scene Management), and cloud (i.e., contains Terrain Management with Terrain DB, and Arena Management). At the device layer, the trainee's interaction with the cockpit is relayed to the Flight Dynamics through Interaction Management. At the edge layer, Flight Oracle uses information from Flight Dynamics to predict and pre-fetch rendered terrain imagery from Terrain Management on the cloud. Arena Management is hosted on the cloud to keep track of all aircraft currently deployed in a given group training simulation. This component maintains an up-to-date view of all aircraft positions and trajectories and relays this information as appropriate to relevant Scene Management. Scene Management processes background scenery with foreground objects before sending them to View Management for the end user. Detail descriptions of those components can be found in Deliverable D1.2.

Before deploying Collins-APP on the CHARITY platform, each of its services is equipped with an interface in order to allow 'Monitoring Agents' to monitor and access the performance of a service. Performance metrics to be monitored include latency, frame rate, bandwidth, computational resources, etc. The domain-specific AEs that are dedicated to each service ingest the monitored data and provide insights which are fed to E2E AE to predict the state of the application. As the use case owner, Collins defines its application QoE requirements which help to identify when its application degrades and the logic how its application should be adapted to maintain the QoE by E2E DE, and mitigation actions should be performed. For example, instead of using a service instance that generates high resolution of imagery, the application switches to another service instance that generates lower resolution images due to low bandwidth issue.

Collins also defines a blueprint that describes how all the components relate to each other. This resulting blueprint, containing the description of all necessary functional & non-functional components as well as the topology of different components, will be uploaded to the 'XR Service Blueprint Template Repository'.

4.2.7.2 Application Deployment

In this phase, Collins-XR is deployed on the CHARITY platform. A blueprint from the 'XR Service Blueprint Template Repository' is selected and customized (if needed, e.g., constraints on services) before sending to CHARITY for deployment. CHARITY takes care of scheduling the deployment of XR services based on received blueprint and the information of end users by using its the 'XR Service Enabler Repository', 'Resource Planning', 'Resource Indexing', and 'Plane Services Registry & Discovery'.

The CHARITY orchestrator deploys the components at the optimal domains and stitch these domains together to offer seamless connectivity between the components. Upon creation, the different components will register themselves in the 'Plane Services Registry & Discovery'. The registered components will use this plane to connect to each other. Once all services are up and running, the end users can begin using Collins-XR and Monitoring Agents start monitoring those services.

4.2.7.3 Application Exploitation

At the exploitation phase, the trainees would be performing a XR collaborative flight training. During this phase, CHARITY would be responsible for ensuring the satisfaction of the KPIs, meaning that all closed loops would be functional. In what follows, we will consider some hypothetical scenarios to show the interplay between the different components.

- Consider a flight simulation is ongoing and a new trainee request to join in the same simulation. A local session in user device will be started and sends the request Collins-APP. This request will be forwarded to CHARITY for resource provision. CHARITY will configure and deploy app components in an edge cloud close to the new trainee's location. Once the app components are up and running, they register themselves in the 'Plane Services Registry &

Discovery' and Arena Manager.

- After a while monitoring, E2E AE detects or predicts that there is an overload or congestion in one or more services which can degrade the QoE. It will notify the Decision Engine. This component will decide which action it should take to solve the issue. For example, the action could be allocating more resources for the issuing service (e.g., Scene Management) which will solve the issue for a time.
- After a while, the AE predicts that the bandwidth between cloud and edge decreases which would degrade the QoS of end users. The alerted is raised and DE decides to reduce the frame rate in Terrain Management and increase level of frame interpolation in Scene Management.

4.2.7.4 Application Decommissioning

Finally, during this phase, the resources are reclaimed. The blueprint inside the 'Running XR Services Repository' is marked as 'terminating' and it is deleted once all the related components are decommissioned. Once the Collins-XR application is terminated, the related shared services are inspected to be also decommissioned. If the shared services are not being used by other XR services, they are terminated. Otherwise, they will be kept, but the closed loops inside them will eventually reduce the amount of committed resources to those services.

5 Conclusions

This deliverable concludes the work carried out within Task 1.2 of the project. The task finishes by M12, i.e., Dec. 2021. The outcome of the research activities is the design of the CHARITY architecture as introduced in Section 3. The architecture will be used as a reference for all tasks of WP2 and WP3, as well as for the integration activities of WP4 (Figure 54). Regarding the latter, this deliverable has already identified the potential open sources and tools that can be leveraged to implement the architecture. A thorough comparison among these tools and the selection of the right ones define one of the future research directions of WP4.

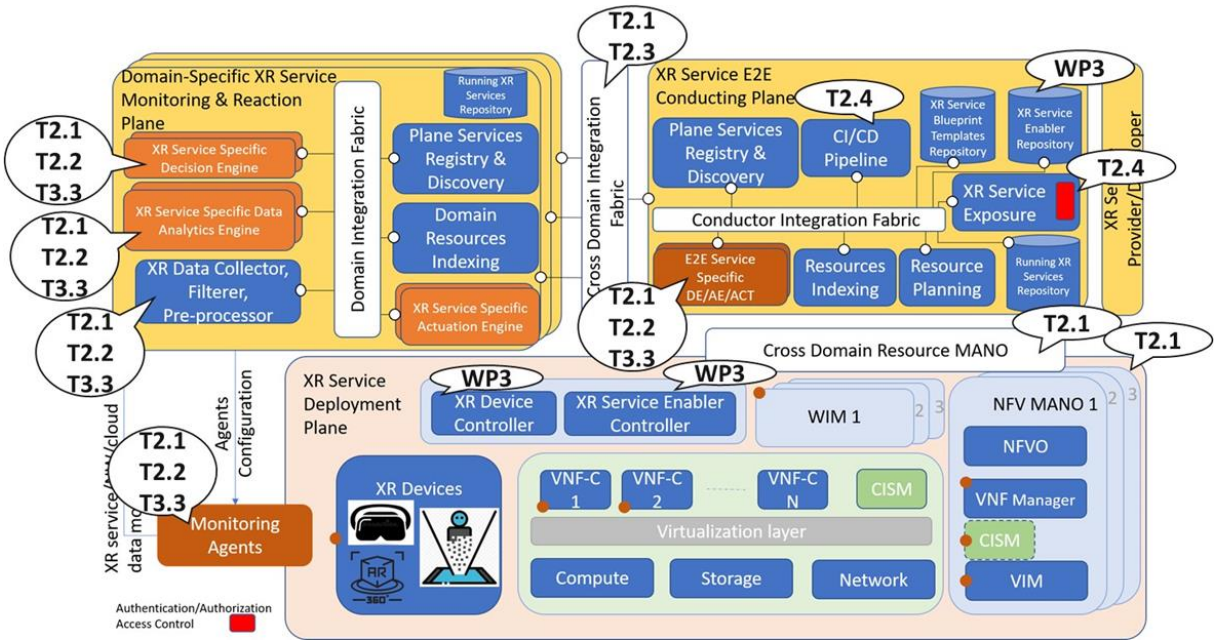

Figure 54: CHARITY architecture and project task mapping

References

[1] ETSI, M. (2019). Multi-access edge computing (MEC) framework and reference architecture. ETSI GS MEC, 3, V2.

[2] https://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/02.02.01_60/gs_MEC003v020201p.pdf

[3] ETSI, M. (2018). Phase 2: Use Cases and Requirements. ETSI GS MEC, 2, V2.

[4] Hu, Y. C., Patel, M., Sabella, D., Sprecher, N., & Young, V. (2015). Mobile edge computing—A key technology towards 5G. ETSI white paper, 11(11), 1-16.

[5] Contreras, L. M., & Bernardos, C. J. (2020). Overview of architectural alternatives for the integration of ETSI MEC environments from different administrative domains. Electronics, 9(9), 1392.

[6] ETSI GR MEC 035 V3.1.1 (2021-06): "Multi-access Edge Computing (MEC); Study on Inter-MEC systems and MEC-Cloud systems coordination.

[7] Sun, Y., Chen, Z., Tao, M., & Liu, H. (2019). Communications, caching, and computing for mobile virtual reality: Modeling and tradeoff. IEEE Transactions on Communications, 67(11), 7573-7586.

[8] Taleb, T., Samdanis, K., Mada, B., Flinck, H., Dutta, S., & Sabella, D. (2017). On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration. IEEE Comm. Surveys & Tutorials, 19(3), 1657-1681.

[9] Pham, Q. V., Fang, F., Ha, V. N., Piran, M. J., Le, M., Le, L. B., ... & Ding, Z. (2020). A survey of multi-access edge computing in 5G and beyond: Fundamentals, technology integration, and state-of-the-art. IEEE Access, 8, 116974-117017.

[10] Jiang, X., Yu, F. R., Song, T., & Leung, V. C. (2021). A Survey on Multi-Access Edge Computing Applied to Video Streaming: Some Research Issues and Challenges. IEEE Communications Surveys & Tutorials, 23(2), 871-903.

[11] ETSI GS NFV-IFA 040 V4.1.1 (2020-11)

[12] NFV ISG. Network Function Virtualisation Infrastructure Architecture - Overview. Technical report, 2015.

[13] ETSI GS NFV-002 V1.1.1 (2013-10) - https://www.etsi.org/deliver/etsi_gs/nfv/001_099/002/01.01.01_60/gs_nfv002v010101p.pdf

[14] ETSI GS NFV-INF 003 V1.1.1 (2014-12)

[15] ETSI GS NFV-INF 004 V1.1.1 (2015-01)

[16] ETSI GS NFV-INF 005 V1.1.1 (2014-12)

[17] ETSI GS NFV-MAN 001 V1.1.1 (2014-12)

[18] ETSI GS NFV-IFA 009 V1.1.1 (2016-07)

[19] Kekki, Sami, Walter Featherstone, Yonggang Fang, Pekka Kuure, Alice Li, Anurag Ranjan, Debashish Purkayastha et al. "MEC in 5G networks.", ETSI white paper 28 (2018): 1-28.

[20] 3GPP, "Release 17" [Online]. Available: https://www.3gpp.org/release-17

[21] 3GPP, "TR 23.758 Study on application architecture for enabling Edge Applications"

[22] 3GPP, "TS 23.558 Architecture for enabling Edge Applications"

[23] 3GPP, "TR 23.748 Study on enhancement of support for Edge Computing in 5G Core network (5GC)"

[24] 3GPP, "TR 33.839 Study on security aspects of enhancement of support for edge computing in 5G Core"

[25] 3GPP, "TR 26.803 Study on 5G Media Streaming Extensions for Edge Processing"

[26] 3GPP, "TR 28.814 Study on enhancements of Edge Computing management"

[27] ETSI, "Harmonizing standards for edge computing - A synergized architecture leveraging ETSI ISG MEC and 3GPP specifications", July 2020.

[28] O. N. Foundation, "CORD" [Online]. Available: https://opennetworking.org/cord/.

[29] M. Consortium, "LL-MEC" [Online]. Available: https://mosaic5g.io/ll-mec/.

[30] E. Coronado, Z. Yousaf and R. Riggio, "lightedge, a lightweight, ETSI-compliant MEC solution for 4G and 5G networks."

[31] Confais, Bastien, Adrien Lebre, and Benoît Parrein. "Performance analysis of object store systems in a fog and edge computing infrastructure." Transactions on Large-Scale Data-and Knowledge-Centered Systems XXXIII. Springer, Berlin, Heidelberg, 2017. 40-79.

[32] Clarke, Ian, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. "Freenet: A distributed anonymous information storage and retrieval system." In Designing privacy enhancing technologies, pp. 46-66. Springer, Berlin, Heidelberg,

2001.

[33] Gheorghe, Alin-Gabriel, Constantin-Cosmin Crecana, Catalin Negru, Florin Pop, and Ciprian Dobre. "Decentralized Storage System for Edge Computing." In 2019 18th International Symposium on Parallel and Distributed Computing (ISPDC), pp. 41-49. IEEE, 2019.

[34] Sonbol, Karim, Öznur Özkasap, Ibrahim Al-Oqily, and Moayad Aloqaily. "EdgeKV: Decentralized, scalable, and consistent storage for the edge", Journal of Parallel and Distributed Computing 144 (2020): 28-40.

[35] Lujic, Ivan, Vincenzo De Maio, and Ivona Brandic. "Efficient edge storage management based on near real-time forecasts." In 2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC), pp. 21-30. IEEE, 2017.

[36] Xing, Jiarong, Hongjun Dai, and Zhilou Yu. "A distributed multi-level model with dynamic replacement for the storage of smart edge computing.", Journal of Systems Architecture 83 (2018): 1-11.

[37] Yang, Ruizhe, F. Richard Yu, Pengbo Si, Zhaoxin Yang, and Yanhua Zhang. "Integrated blockchain and edge computing systems: A survey, some research issues and challenges.", IEEE Communications Surveys & Tutorials 21, no. 2 (2019): 1508-1532.

[38] Huang, Yaodong, Xintong Song, Fan Ye, Yuanyuan Yang, and Xiaoming Li. "Fair and efficient caching algorithms and strategies for peer data sharing in pervasive edge computing environments.", IEEE Transactions on Mobile Computing 19, no. 4 (2019): 852-864.

[39] Hou, Tingting, Gang Feng, Shuang Qin, and Wei Jiang. "Proactive content caching by exploiting transfer learning for mobile edge computing.", International Journal of Communication Systems 31, no. 11 (2018): e3706.

[40] Chang, Zheng, Lei Lei, Zhenyu Zhou, Shiwen Mao, and Tapani Ristaniemi. "Learn to cache: Machine learning for network edge caching in the big data era.", IEEE Wireless Communications 25, no. 3 (2018): 28-35.

[41] Zhang, Li, Jun Wu, Shahid Mumtaz, Jianhua Li, Haris Gacanin, and Joel JPC Rodrigues. "Edge-to-edge cooperative artificial intelligence in smart cities with on-demand learning offloading." In 2019 IEEE Global Communications Conference (GLOBECOM), pp. 1-6. IEEE, 2019.

[42] Ale, Laha, Ning Zhang, Huici Wu, Dajiang Chen, and Tao Han. "Online proactive caching in mobile edge computing using bidirectional deep recurrent neural network.", IEEE Internet of Things Journal 6, no. 3 (2019): 5520-5530.

[43] ETSI, GS ZSM 002 Zero-touch network and Service Management (ZSM); Reference Architecture.

[44] ETSI, GS ZSM 001 Zero Touch Network and Service Management (ZSM); Requirements Based on Documented Scenarios;

[45] ETSI, GS ZSM 009-1 V1.1.1 (2021-06) Zero-touch network and Service Management (ZSM); Closed-Loop Automation; Part 1: Enablers

[46] ETSI TS 138 401 V15.5.0 (2019-05), "5G; NG-RAN; Architecture description"

[47] S. Niknam, A. Roy, H. S. Dhillon, S. Singh, R. Banerji, J. H. Reed, N. Saxena and S. Yoon, "Intelligent O-RAN for Beyond 5G and 6G Wireless Networks", https://arxiv.org/abs/2005.08374, May 2020.

[48] A. Weissberger, "TIP OpenRAN and O-RAN Alliance liaison and collaboration for Open Radio Access Networks", Feb 2020. [Online]. Available: https://techblog.comsoc.org/2020/02/26/tip-openran-and-o-ran-alliance-liaison-and-collaboration-for-open-radio-access-networks/.

[49] O. R. P. Coalition, "Promoting the Deployment of 5G Open Radio Access Networks", May 2021. [Online]. Available: https://www.openranpolicy.org/wp-content/uploads/2021/06/ORPC-Open-RAN-NOI-Reply-Comment-Letter-as-filed-May-28-2021-c3.pdf.

[50] Nokia, "Open RAN," [Online]. Available: https://www.nokia.com/networks/portfolio/radio-access-networks-ran/open-ran/.

[51] F. J. L. Hernando, E. S. Santiago, M. A. Peña, A. C. Ricciulli and J. L. E. Gutiérrez, "Telefónica views on the design, architecture, and technology of 4G/5G Open RAN networks", January 2021. [Online]. Available: https://www.telefonica.com/documents/737979/145981257/Whitepaper-OpenRAN-Telefonica.pdf.

[52] N. Docomo, "5G Open RAN Ecosystem Whitepaper," June 2021. [Online]. Available: https://www.nttdocomo.co.jp/binary/pdf/corporate/technology/whitepaper_5g_open_ran/OREC_WP.pdf.

[53] Ericsson, "Openness, innovation and flexibility: Open RAN explained," [Online]. Available: https://www.ericsson.com/en/openness-innovation/open-ran-explained.

[54] O.-R. S. community, "Cherry release," Dec 2020. [Online]. Available: https://wiki.o-ran-sc.org/pages/viewpage.action?pageId=20876303.

[55] O. N. Foundation, "Converged Multi-Access and Core," [Online]. Available: https://opennetworking.org/comac/.

[56] O. Sunay, T. Vachuskal and S. Das, "Open Networking Foundation: SD-RAN," [Online]. Available: https://opennetworking.org/sd-ran/.

[57] Federico Chiariotti, Stepan Kucera, Andrea Zanella, and Holger Claussen. "LEAP: A latency control protocol for multi-path data delivery with pre-defined QoS guarantees." In IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 166-171. IEEE, 2018.

[58] Sangtae Ha, Injong Rhee, and Lisong Xu. "CUBIC: a new TCP-friendly high-speed TCP variant." ACM SIGOPS operating systems review 42, no. 5 (2008): 64-74. DOI: 10.1145/1400097.1400105

[59] Mach Chen, Xuesong Geng, and Zhenqiang Li. "Segment Routing (SR) Based Bounded Latency" IETF. draft-chen-detnet-sr-based-bounded-latency-01. 2019.

[60] 3GPP TS 26.501 V16.6.1 (2021-01), "5G Media Streaming (5GMS); General description and architecture (Release 16)"

[61] 3GPP TR 26.918 V16.0.0 (2018-12), "Virtual Reality (VR) media services over 3GPP (Release 16)". In a discussion of VR service architectures in referenced document, the authors confine themselves to addressing the scenario of "VR content in file or segment based download and streaming services".

[62] 3GPP TS 26.118 V16.1.0 (2020-12), "Virtual Reality (VR) profiles for streaming applications (Release 16)". Similarly in this referenced document, the authors only consider the distribution of VR content "in file-based download and DASH-based streaming services". A closed loop, real-time VR streaming use case has not yet been addressed by 3GPP

[63] 3GPP TS 26.118 V16.1.0 (2020-12), "Virtual Reality (VR) profiles for streaming applications (Release 16)". Similarly in this referenced document, the authors only consider the distribution of VR content "in file-based download and DASH-based streaming services". A closed loop, real-time VR streaming use case has not yet been addressed by 3GPP

[64] ISO/IEC 18039:2019 Information technology — Computer graphics, image processing and environmental data representation — Mixed and augmented reality (MAR) reference model

[65] Lee, J., Lee, Y., Lee, S., & Kim, G. J. (2013, November). Standardization for augmented reality: introduction of activities at ISO-IEC SC 24 WG 9. In Proceedings of the 12th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry (pp. 279-280).

[66] ISO/IEC 23000-13:2017(en) Information technology - Multimedia application format (MPEG-A) — Part 13: Augmented reality application format

[67] Nokia OMAF Implementation, Mar. 2021, [online] Available: https://github.com/nokiatech/omaf

[68] T. Huang and Y. Liu, "3d point cloud geometry compression on deep learning," presented at the Proceedings of the 27th ACM International Conference on Multimedia, 2019, pp. 890–898.

[69] D. C. Garcia and R. L. de Queiroz, "Intra-Frame Context-Based Octree Coding for Point-Cloud Geometry," presented at the 2018 25th IEEE International Conference on Image Processing (ICIP), 2018, pp. 1807–1811.

[70] T. Wiemann, F. Igelbrink, S. Pütz, M. K. Piening, S. Schupp, S. Hinderink, J. Vana, and J. Hertzberg, "Compressing ROS sensor and geometry messages with Draco," presented at the 2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), 2019, pp. 243–248.

[71] A. Varischio, F. Mandruzzato, M. Bullo, M. Giordani, P. Testolina, and M. Zorzi, "Hybrid Point Cloud Semantic Compression for Automotive Sensors: A Performance Evaluation," arXiv preprint arXiv:2103.03819, 2021.

[72] M. Hosseini and C. Timmerer, "Dynamic adaptive point cloud streaming," presented at the Proceedings of the 23rd Packet Video Workshop, 2018, pp. 25–30.

[73] X. Sun, S. Wang, M. Wang, S. S. Cheng, and M. Liu, "An Advanced LiDAR Point Cloud Sequence Coding Scheme for Autonomous Driving," presented at the Proceedings of the 28th ACM International Conference on Multimedia, 2020, pp. 2793–2801.

[74] L. Wiesmann, A. Milioto, X. Chen, C. Stachniss, and J. Behley, "Deep Compression for Dense Point Cloud Maps," IEEE Robotics and Automation Letters, vol. 6, no. 2, pp. 2060–2067, 2021.

[75] B. Han, Y. Liu, and F. Qian, "ViVo: Visibility-aware mobile volumetric video streaming," presented at the Proceedings of the 26th Annual International Conference on Mobile Computing and Networking, 2020, pp. 1–13.

[76] C. Portaneri, P. Alliez, M. Hemmer, L. Birklein, and E. Schoemer, "Cost-driven framework for progressive compression of textured meshes," presented at the Proceedings of the 10th ACM Multimedia Systems Conference, 2019, pp. 175–188.

[77] K. Christaki, E. Christakis, P. Drakoulis, A. Doumanoglou, N. Zioulis, D. Zarpalas, and P. Daras, "Subjective Visual Quality Assessment of Immersive 3D Media Compressed by Open-Source Static 3D Mesh Codecs," Cham, 2019, pp. 80–91.

[78] ETSI GS ARF 003 - V1.1.1

[79] An overview of ongoing point cloud compression standardization activities: video-based (V-PCC) and geometry-based (G-PCC) https://www.cambridge.org/core/journals/apsipa-transactions-on-signal-and-information-processing/article/an-overview-of-ongoing-point-cloud-compression-standardization-activities-videobased-vpcc-and-geometrybased-gpcc/56FCAF660DD44348BCB1BCA9B5EC56CF

[80] ANASTACIA. "Final Architecture Design"

[81] INSPIRE-5Gplus. "D2.2: Initial Report on Security Use Cases, Enablers and Mechanisms for Liability-aware Trustable Smart 5G Security"

[82] Chafika Benzaid, Tarik Taleb, Cao-Thanh Phan, Christos Tselios, George Tsolis. Distributed AI-based Security for Massive Numbers of Network Slices in 5G & Beyond Mobile Systems. In Proc. of European Conference on Networks and Communications (EuCNC) – 6G Summit, Virtual Conference (Porto, Portugal), 8 – 11 June, 2021.

[83] ETSI GR NFV-IFA 029. Network Functions Virtualisation (NFV) Release 3; Architecture; Report on the Enhancements of the NFV architecture towards “Cloud-native” and “PaaS”. V3.3.1, Nov 2019.

[84] CHARITY. “D1.2: Reference scenarios, use cases and requirements”

