HEXA-X-II

A holistic flagship towards the 6G network platform and system, to inspire digital transformation, for the world to act together in meeting needs in society and ecosystems with novel 6G services

Deliverable D2.4
End-to-end system evaluation results from the interim overall 6G system

Co-funded by the European Union

6GSNS

Hexa-X-II project has received funding from the Smart Networks and Services Joint Undertaking (SNS JU) under the European Union's Horizon Europe research and innovation programme under Grant Agreement No 101095759.

Date of delivery:	30/09/2024	Version:	1.0
Project reference:	101095759	Call:	HORIZON-JU-SNS-2022
Start date of project:	01/01/2023	Duration:	30 months

Document properties:

Document Number:	D2.4
Document Title:	End-to-end system evaluation results from the interim overall 6G system.
Editor(s):	Pol Alemany and Behnam Ojaghi (CTTC)
Authors:	Raul Muñoz, Ricard Vilalta (CTT), Sylvaine Kerboeuf (NFR), Markus Staufer (NGE), Pawani Porambage, Dinaj Attanayaka, Rafael Pires, Henry Blue, Jere Malinen, Yasintha Rumesh (VTT), Betul Guvenc Paltun, Ferhat Karakoc (EBY), Anton Schösser (TUD), Sonika Ujjwal (LMF), Prajnamaya Dass, Ali Khandan Boroujeni, Stefan Köpsell (BI), Diego Lopez, Antonio Pastor, Jose M. Manjon, Javier Velazquez (TID), Christina Karousatou, Sokratis Barmpounakis, Panagiotis Demestichas (WIN), Heikki Karvonen, Jukka-Pekka Salminen (SIS), Ioannis Tzanettis, Grigoris Kakkavas, Anastasios Zafeiropoulos (ICC), Pietro G. Giardina, Michael De Angelis, Giada Landi (NXW), Selim Ickin, Hannes Larsson (EAB)
Contractual Date of Delivery:	30/09/2024
Dissemination level:	PU
Status:	Final
Version:	1.0
File Name:	Hexa-X-II D2.4

Revision History

Revision	Date	Issued by	Description
0.1	31.01.2024	Hexa-X-II WP2	Template for Deliverables/IRs
0.5	18.06.2024	Hexa-X-II WP2	First contribution for internal review.
0.7	12.07.2024	Hexa-X-II WP2	Internal Review
0.8	30.08.2024	Hexa-X-II WP2	External Review
0.9	20.09.2024	Hexa-X-II WP2	GA Review
1.0	30.09.2024	Hexa-X-II WP2	Final version submitted to EC approval

Abstract

This report is focused on a set of experimental and simulation results towards a complete evaluation of the E2E System through the second iteration of the System-PoC (i.e., System-PoC #B). Moreover, this document also presents the results associated to different security aspects around needed to achieve a high-level of protection of the E2E System.

Keywords

6G, design principles, E2E System evaluation, experimental results, Key Performance Indicator (KPI), Key Value Indicator (KVI), Management & Orchestration, privacy, Proof of Concept, resilience, security, simulations, sustainability, System-PoC

Executive Summary

An overview of the comprehensive 6G end-to-end (E2E) system design, elaborated in D2.3 [HEX224-D23], focuses on the key innovations within its elements and the various layers that constitute it. This design aims to address the challenges of the future network platform, extending 6G services beyond communications to applications, ecosystems, verticals, and users. Its goal is to create value for society, the economy, and the environment.

To validate the system design outlined by the current 6G blueprint, we must evaluate several critical aspects against the design principles defined in D2.1. These principles include supporting and exposing 6G services, increasing the level of automation, ensuring flexibility, scalability, resilience, availability, security, privacy, cloud-optimized internal interfaces, and environmental sustainability.

Insight into how the current system design aligns with the 6G system design principles is provided. This assessment is based on the evaluation results from System-PoC #B and considerations related to security, privacy, and system-level resilience results.

Significant advancements in managing and orchestrating 6G services can be achieved by integrating intent-based network solutions, distributed and federated orchestration across the cloud continuum including devices, closed-loop automation, and AI-assisted lifecycle management. The trustworthy flexible topology minimizes interruption time, enhances availability, and ensures service continuity. Additionally, innovative 6G services like compute offloading contribute to energy savings within devices. The management capability exposure framework facilitates secure, privacy-preserving, and efficient data and service capability exposure.

Results for the evaluation of the proposed approach to trustworthiness, via Security, Privacy and Resilience (SPR) enablers, are provided, according to their Technology Readiness Levels (TRL), together with the integration patterns already identified for the most mature of those enablers. These integration patterns take into account the main security features analysed within the project at large, and address the enablers identified as relevant for security management among other SNS JU projects. The enablers with a lowest TRL correspond to physical layer features, and the outcomes of conceptual evaluation by means of simulation or experimental laboratory environments are provided. With an intermediate TRL come those associated with the trust fabric future network services will rely on, including trustworthy AI, the use of simulation environments, and the lifecycle management of security controls, and this document addresses the experimental environments used to evaluate them and the corresponding enabler interfaces. Finally, the results of initial experiments evaluating the conditions and cases for the application of the highest TRL solutions are considered, together with the applicable control interfaces and considerations on their use.

The evaluation results confirm that the current system design principles set forth in D2.1 are effectively met, providing a robust foundation for future 6G developments.

Table of Contents

List of Tables

List of Figures

Acronyms and abbreviations

Term	Description
AI	Artificial Intelligence
AIaaS	AI-as-a-Service
AIaaSF	AI-as-a-Service Function
AMF	Access and Mobility Management Function
AMR	Autonomous Mobile Robot
aPGD	Auto-PGD
API	Application Programming Interface
AUSF	Authentication Server Function
AWGN	Additive White Gaussian Noise
CAPEX	Capital Expenses
CDF	Cumulative Distribution Function
CIR	Channel Impulse Response
CL	Closed-Loop
CN	Core Network
CNF	Core Network Function
CNI	Container Network Interface
CNN	Convolutional neural Networks
Cobot	Collaborative Robot
CP	Control Plane
CPU	Central Processing Unit
CSG	Cell Site Gateways
CU	Central Unit
DataOps	Data Operations
DDoS	Distributed Denial of Service
DLT	Distributed Ledger Technology
DT	Digital Twin
DU	Distributed Unit
DWDM	Dense Wavelength-Division Multiplexing
ED	Energy Detector

eMBB	enhanced Mobile Broadband
EPC	Enclave Page Cache
E2E	End-to-End
FGSM	Fast Gradient Sign Method
FL	Federated Learning
FTN	Flexible Topology Node
GDPR	General Data Protection Regulation
gMNI	gRPC Network Management Interface
gRPC	google Remote Procedure Call
HLS	Higher Layer Split
IBE	Intent-based Entity
IBM	Intent-based Management
IBN	Intent-based Network
IDS	Intrusion Detection System
IETF	Internet Engineering Task Force
ILE	Infrastructure Layer Emulator
ISO	International Organization for
JCAS	Joint Communication and Sensing
KGR	Key Generation Rate
KMS	Key Management Service
kNN	k-Nearest Neighbours
KPI	Key Performance Indicator
KVI	Key Value Indicator
LFP	Leakage-Failure Probability
LLS	Lower Layer Split
LOF	Local Outlier Factor
LoS	Line of Sight
LoT	Level of Trust
LoTAF	Level of Trust Assessment Function
LSTM	Long Short-Term Memory
MAC	Medium Access Control

MCEF	Management Capabilities Exposure Framework
ML	Machine learning
MLOps	Machine Learning Operations
mMTC	massive Machine Type Communication
mTLS	mutual Transport Layer Security
MTTF	Mean Time to Failure
M&O	Management and Orchestration
NBI	North-bound Interface
NLP	Natural Language Processing
NN	Neural Network
NSM	Network Service Mesh
NWDAF	Network Data Analytics Function
OAI	Open Air Interface
OFDM	Orthogonal Frequency Domain Multiplexing
OLS	Open Line System
OPEX	Operating Expenses
OPoT	Ordered Proof of Transit
O-RAN	Open RAN
PCA	Principal Component Analysis
PCF	Policy Control Function
PDCP	Packet Data Convergence Protocol
PGD	Projected Gradient Decent
PHY	Physical
PLD	Physical Layer Deception
PLS	Physical Layer Security
PoC	Proof-of-Concept
PQC	Post-Quantum Cryptography
PT	Physical Twin
QoE	Quality of Experience
QoS	Quality of Service
RAN	Radio Access Network

REC-EXEC	Resource Orchestrator
RIC	RAN Intelligent Controller
RL	Reinforcement Learning
RLC	Radio Link Control
ROADM	Reconfigurable Optical Add-Drop Multiplexer
ROC	Receiver Operating Characteristics
ROS	Robot Operating System
RPC	Remote Procedure Call
RRC	Radio Resource Control
RTMP	Real Time Messaging Protocol
RU	Radio Unit
SA	Stand Alone
SCF	Sensing Control Function
SBI	South-bound Interface
SeMF	Sensing Management Function
SFC	Service Function Chain
SLA	Service level Agreement
SLO	Service Level Objectives
SPCTM	Sensing Policy, Consent, and Transparency Management
SPF	Sensing Processing Functions
SPR	Security, privacy and Resilience
SU	Sensing Unit
SM	Support Vector Machine
SXC	Spatial Cross Connect
TAPI	Transport API
TDD	Time Division Duplexing
TEE	Trusted Execution Environment
TFS	TeraFlowSDN
TLS	Transport Layer Security
TPS	Transactions per Second
TX	Transmitter

UAV	Unmanned Aerial Vehicle
UC	Use Case
UDM	Unified Data Management
UDP	User Datagram Protocol
UDR	Unified Data Repository
UE	User Equipment
UP	User Plane
UPF	User Plane Function
URLLC	Ultra-Reliable Low-Latency Communication
VM	Virtual Machine
VNF	Virtual Network Function
WebUI	Web User Interface
WN	Worker Node
WP	Work Package
XAI	Explainable AI

1 Introduction

Hexa-X-II is the 6G Flagship project under the European Union Horizon Europe research and innovation program Smart Network and Services Joint Undertaking (SNS JU). This document is the fourth public deliverable of Work Package 2 (WP2) – End-to-end (E2E) system evaluation results from the interim overall 6G system. The outcomes presented in this deliverable have a strong relationship with the works done in the previous deliverables D2.1 [HEX223-D21], D2.2 [HEX223-D22] and D2.3 [HEX224-D23].

This document aims to become a transitional outcome within the Hexa-X-II lifetime. On the one hand, this deliverable aims to describe and illustrate the set of results and outcomes originated from all the work presented in the previous deliverables, and on the other hand, this deliverable will allow to trigger the last phase of the 6G E2E System blueprint study and the System-PoC implementations work that will lead towards the future publication of the last two WP2 deliverables (i.e., D2.5 and D2.6), which will present the last set of outcomes produced by the Hexa-X-II project. The Hexa-X-II system blueprint design, coupled with the validation of its technology enablers through system-PoCs, will provide suitable inputs for impacting the global view 6G on harmonization and standardisation. This influence is already reflected in D7.5 [HEX223-D7.5]. The project's timeline aligns perfectly with the global 6G standardisation horizon, e.g. the first 6G specifications will be delivered in 3GPP release 21 by the end of 2028, building upon 6G studies in release 20, which recently started and will conclude in early 2027.

On what regards the System-PoC, the current deliverable may be identified as the milestone that determines the end of the System-PoC #B working period, which allows to evolve the focus of the whole System-PoC. This evolution started with the study of smart network management aspects for demonstrating management mechanisms (i.e., System-PoC #A). Then, its scope increased with the study of network architecture elements and the refinements of management (i.e., System-PoC #B) based on the outcomes presented in this document. Finally, the work presented in this deliverable will trigger the last System-PoC phase with the addition of radio and devices aspects and refinements of 6G architecture design and smart network management (i.e., System-PoC #C).

1.1 Objective of the document

The objectives of this document can be identified with the following three elements:

- **Objective 1:** To introduce the current 6G E2E system design status with an overview of its elements and the different layers composing it, together with those aspects that should be evaluated and validated.
- **Objective 2:** To provide the next round of experimental outcomes obtained from an E2E system point of view. This objective is divided in two sub-objectives:
  - Sub-Objective 2.1: To present the experimental results related to the System-PoC #B and its components associated. In this deliverable a clear evolution is presented since the outcomes illustrated in D2.3 [HEX224-D23].
  - Sub-Objective 2.2**:** To present the results achieved by the set of security-related enablers introduced in D2.3 [HEX224-D23] with the focus of enforcing security on 6G E2E systems.
- **Objective 3:** To conclude how the different experiments and their results allow to validate the different aspects presented in objective 1.

1.2 Structure of the document

This document is structured as follows: Chapter 2 provides a brief introduction and description of the latest E2E System design and the aspects under evaluation. Chapter 3 presents the first set of the E2E system-level evaluation results with special focus on the System-PoC B; the designed architecture with its components and the implemented scenarios used to obtain the experimental outcomes. Chapter 4 describes the obtained experimental outcomes from the security enablers presented in Hexa-X-II deliverable D2.3 [HEX224-D23], bringing topics such as anomaly detection, Joint Communication and Sensing (JCAS) threat mitigation, trust aspects and the use of Artificial Intelligence (AI). Finally, Chapter 5 presents the conclusions of this document, with special focus on the 6G E2E System evaluation aspects based on the presented results.

2 Introduction to E2E System

This chapter provides a summary of the 6G system design process presented in [HEX223-D21], refined and applied in both [HEX223-D22] and [HEX224-D23]. The latest version of the E2E 6G system blueprint is presented as reference in section 2.1.

2.1 E2E system design overview

The 6G E2E system blueprint defined by Hexa-X-II is built upon an iterative methodology. Enablers t developed in the technical work packages of the project focus on specific aspects of 6G system. Enablers are further analysed and selected for their integration in the E2E 6G system design considering the system requirements defined from the 6G use-cases, including the objectives related to the sustainability aspects from [HEX223-D12]. Consequently, an overall interim design of the 6G system blueprint is elaborated, where the selected enablers are mapped onto this blueprint. The mapping captures the dependence between the different enablers and the different layers (application, application enablement platform, network function, infrastructure) and with the pervasive functionalities. The system blueprint design is further evaluated via system-level proof of concepts or simulation-based approaches to estimate (Key Performance Indicators (KPIs) and Key Value Indicators (KVIs) where the results can be fed back to technical work packages for iteratively improving the system design.

Following this iterative design process, the interim 6G E2E system blueprint was defined in [HEX224-D23], see Figure 2-1. The blueprint consists of four layers (application, application enablement platform, network functions, and infrastructure layers) and pervasive functionalities that can reside in any of those layers as well as the corresponding interfaces towards their interactions.

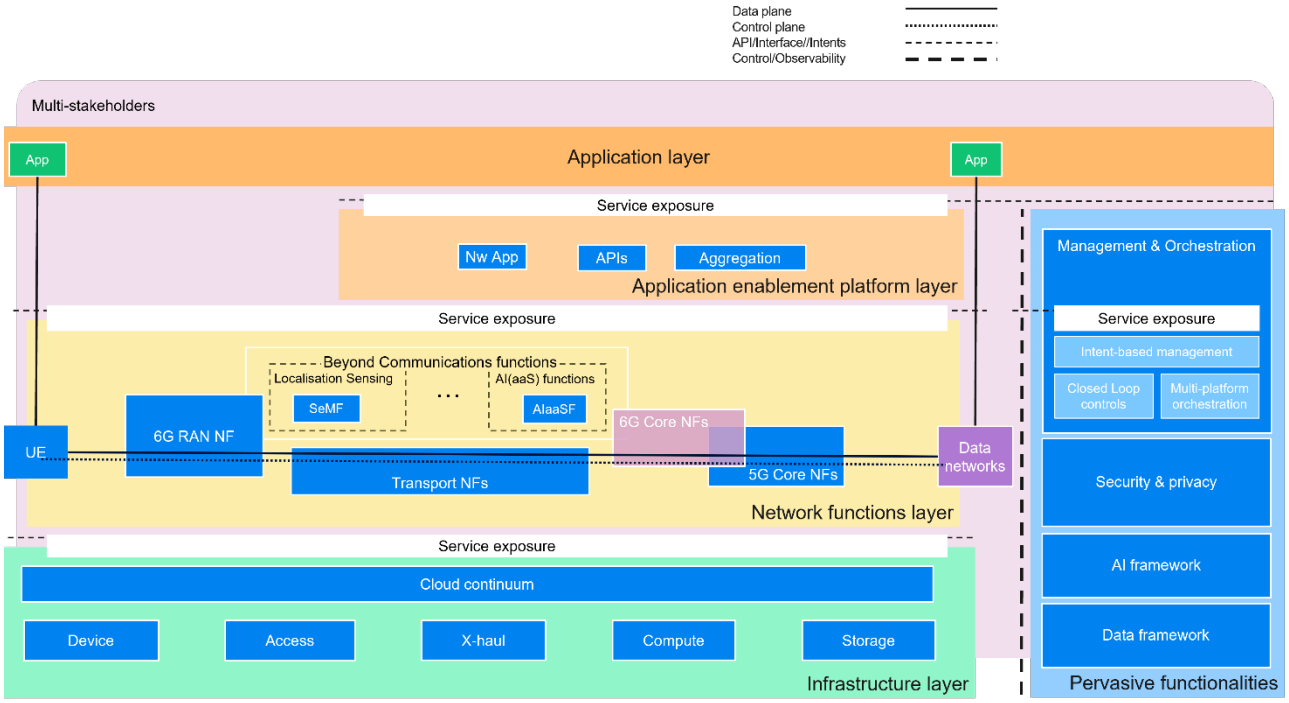

Figure 2-1: 6G E2E system blueprint from [HEX224-D23].

The main aspects of the 6G E2E system blueprint are summarized as follows:

- The infrastructure layer encompasses all the E2E infrastructure (device, access, x-haul, compute, and storage blocks). It provides network and compute resources over which the various functionalities of the 6G system are executed and orchestrated. It introduces the **cloud continuum** principle, which will be essential for the 6G E2E system as discussed later in the system-PoCs.
- The network function layer comprises beyond communications functions as well as the 6G Radio Access Network (RAN) and core network (CN) functions blocks.

  - Considering the migration principles, the **6G radio access** should support a single-Radio Access Technology (RAT) architecture only, i.e., a 6G User Equipment (UE) that connects via the 6G radio interface establishes a connection to the CN for 6G without any complex inter-RAT multi-connectivity, in contrast to the 5G paradigm.
  - The **6G Core network function** will be an extension of 5G Core network instead of a subset.
  - The **6G Beyond Communications Functions** depicts the functionalities to realize new services expanding beyond the communication capabilities, such as Sensing Management Function (**SeMF**) responsible for facilitating an efficient coordination of sensing procedures, or AIaaS functions (**AIaaSF**) responsible for providing accessible AI capabilities as pre-built AI models, datasets, algorithms, and tools into applications. Other 6G Beyond Communications capabilities will exist such as offering Compute-as-a Service and are not (yet) represented in the current blueprint..
- The application enablement platform layer and the 3rd party application layer facilitate the ecosystem applications to interact with the underlying network. These applications encompass both network applications managed/provided by a Communication Service Provider and applications that belong to vertical industries and have specific network requirements. The Application enablement platform layer is the main access via the service exposure to the services of the 6G system as a platform.
  - The Application Programming Interface (**API) box** represents the abstracted APIs from the network and M&O services providing a further abstraction level to simplify their use for non-experts and augmented with additional services from e.g., the network applications, that can be leveraged by the developer of the applications of the Application layer.
  - **Aggregation** of services, i.e., the action to compose services from a set of capabilities potentially from different 6G service providers, e.g., over multiple networks, is also provisioned in this layer.
- The pervasive functionalities facilitate the four layers of the 6G system blueprint to realize the full potential of the 6G platform, either jointly or independently.
  - Both **data and AI frameworks** highlight the AI-centric approach to the 6G system as compared to the 5G system and integrate respectively Data Operations (DataOps) and ML Operations (MLOps).
  - The management and orchestration framework moves beyond the traditional relevant functions in 5G networks towards a more **intent-based management** approach. Hence, the proposed blueprint introduces the Intent based Management framework as one of the pillars of the 6G management and orchestration framework. The **multi-platform orchestration** functionality provides a unified management and orchestration of network services and network applications over a cloud continuum across multiple domains, owned and administered by different stakeholders, and characterized by underlying heterogeneous technologies platforms. The possibly **AI/ML based closed loop controls** as essential for an increasing level of management and orchestration (M&O automation toward autonomy in the 6G network operations.
  - A broader security and privacy framework for the 6G E2E system is required as against a more localized and domain specific approach on different layers of the 5G system.
- The **service exposure framework** exposes relevant service capabilities APIs to external entities (e.g., other services, application developers, network service developers, service providers, etc.) in a secure, privacy-preserving, and efficient manner that is crucial for the 6G network platform. It provides the necessary means for service APIs discovery, secured access control and enforcement. It is worth emphasizing that service in the service APIs is used in the broadest sense of the term and encompasses data, resource, and control.
- Lastly, there will be multiples interactions between the various sets of stakeholders of the 6G ecosystem. As the market is becoming more disaggregated, multiples stakeholders will be engaged in the value creation of the 6G platform, moving away from a linear value chain toward a multi-sided value chain. This leads to the definition of new roles as already described in [HEX223-D22]. Representation of all possible **multistakeholder interfaces** (*i.e.*, interfaces between domains administrated by distinct stakeholders) would complexify the system blueprint. Instead, it is invoked with the multistakeholder box surrounding the various layers and domains of the 6G platform.

2.2 E2E system design aspects under evaluation

The 6G system design outlined by the system blueprint features ten key system design principles with details provided in Hexa-X-II [HEX223-D21]. This sub-section discusses those design principles that can be assessed at this stage of system-PoC development and on the basis of other evaluation presented in the next chapters. The system aspects under evaluation in this deliverable cover eight design principles out of the ten. The two principles that are not evaluated are principle 8 (separation of concerns of network functions) which refers to the optimized functionality in CN and RAN with bounded context and no duplication, avoiding complex interdependencies and cross-functional signalling and principle 9 (Network simplification in comparison to previous generations) which aims to avoid many standardized deployment options and protocol splits. It must be noted that the enablers matching those two principles are currently not part of the system PoC but are reported in D3.3 [HEX224 D3.3].

The system design aspect in terms of required functionalities, including those for operations, to be evaluated for each of the considered system principle is discussed hereafter. The validation that the system design fulfils the design principles is provided at the end (in section 5.1) after the evaluation results presented in section 3 and 4.

- Principle 1: Support and exposure of 6G services and capabilities

For the design of the system, it encompasses the design of new capabilities for supporting novel services that goes beyond communications, such as compute offloading, sensing, AI-as-a-service etc. It also involves generic and dynamic exposure functionalities, e.g. simplified APIs to expose capabilities to E2E applications facilitating seamless integration of beyond communication network functions and hardware capabilities. Such aspects are covered in the system blueprint through the API box in the application enablement platform layer (building simplified and enriched APIs), the service exposure framework available in several places at the different domains and layers in the blueprint, as well as within the Intent-based management in which intent-based APIs are defined to offer services to 6G customers (e.g. Application developers, enterprises from vertical segments). In-network service exposure is also important with for example the management capabilities exposure framework which offers a service bus for smooth Hexa-X-II capability interoperability. It can be used for example for predictive management & orchestration, e.g. proactive scaling or traffic rerouting actions for dynamic adaptation to changing network conditions and user demands, ensuring optimal performance for real-time services.

- Principle 2: Full automation and optimization

The architecture should support full automation of network and service management operations, utilizing distributed AI/ML agents to manage and optimize the system without human interaction.

For the system design, it requires a system built on a pervasive infrastructure across the compute continuum, to provide seamless service orchestration across diverse scenarios. The 6G E2E system requires to establish a comprehensive and widespread data and analysis framework, complemented by a pervasive AI framework and pervasive service management and orchestration. Such aspects are covered in the system blueprint within the functionalities of the same names. The multi-platform orchestration box accounts for components providing synergetic orchestration mechanisms for the compute continuum. In the system blueprint, the inclusion of data fusion, monitoring and telemetry in data framework as well as AI/ML based algorithms are essential enablers to complement the closed loop control box for real time zero touch automation. This allows for continuous optimization to changing network conditions, ensuring optimal performance for real-time services. Distributed AI/ML agents to optimize the system without human interaction is another key component in the design of the system.

- Principle 3: Flexibility to different network scenarios

The architecture should be designed to support digital inclusion. Addition of service capabilities and new service endpoints can be done at run-time without changes to existing E2E services. The network should support increased application awareness and adaptive quality of service (QoS) / quality of experience (QoE).

Fulfilling this design principle requires establishing a pervasive service management and orchestration framework. Multi-agent system for multi-cluster orchestration is allowing the deployment of services over

multi-cluster environments. Federated as well as decentralized orchestration components can allow dynamic connections to third party networks capabilities, enhancing flexibility to different network scenarios and topologies. They also allow for new service endpoints to be established at run-time.

It also involves components for flexible topologies, implementing network programmability framework, where multiple backhaul connectivity services can be provided to support different network splits and scenarios. It also involves enablers for having application awareness and adaptive QoS and QoE.

- Principle 4: Scalability

Fulfilling this design principle focuses on creating a pervasive service management and orchestration system for e.g., scaling in and out based on mobility and time-varying traffic needs. Components of decentralized orchestrations will support both small and large-scale deployments. Network programmability enabler will help the scalability of optimized transport network functions (e.g., over heterogeneous multi-domain/multi-clouds). This scalability is crucial for handling the dynamic demands of real-time services, ensuring consistent performance even under fluctuating traffic loads.

- Principle 5: Resilience and availability

The architecture should allow mobile network operators (MNOs) to build deployments with high resilience and availability. The architecture shall support separation of control plane (CP) and user plane (UP), and resilient mobility solutions as a method to provide service availability. Furthermore, the architecture should support subnetwork resilience e.g., if a subnetwork loses connectivity it should connect with another subnetwork to remove single point of failures.

The 6G E2E system requires service management and orchestration integrating high resilience and availability mechanisms. It encompasses data analysis, AI integration, and coordination mechanism in ensuring resilience. This resilience and availability are essential for ensuring uninterrupted service delivery, particularly for real-time services that require constant connectivity and low latency.

- Principle 6: Persistent security and privacy

The objective is to establish a comprehensive framework in the 6G E2E system that ensures security and privacy are integrated across all components with the goal of assuring a trustworthy environment. E.g., address current as well as future threats in a resilient manner against attacks and incorporate security fundamentals in its design, inherently support the preservation of privacy, and allow different levels of anonymity for future services.

- Principle 7: Internal interfaces are cloud optimized

For the design of the system, it involves having a cloud-native approach in services development and orchestration, considering separation of concerns among the various layers (network function layer, application enablement platform layer, application layer) and exposure and consumption of APIs for interaction among network and edge/cloud application providers.

10. Principle 10: Minimize environmental footprint and enabling sustainable use cases

This principle aims for E2E orchestration, emphasizing energy-efficient and cost-conscious operations. It involves the implementation of a pervasive data and analysis framework alongside the modularization of network functions. The infrastructure layer in the 6G E2E system should optimize both energy consumption and costs for enhanced sustainability and operational efficiency of the use cases.

Designing the systems based on KPIs and KVIs may involve trade-offs due to potential performance decline from adhering to specific KVIs, necessitating their careful consideration in the iterative design process. Specifically, the evaluation will involve firstly whether the proposed components and enablers conform to the agreed upon KPIs and KVIs corresponding to the use case.

3 E2E system-level evaluation results

This chapter describes the process of the Hexa-X-II system assessment with respect to the proposed KPIs and KVIs, both described in section 3.5. The validation process of the E2E system is driven by the design and implementation of PoCs. Within the scope of the E2E validation, various 6G enablers are assessed, including aspects related to management and orchestration over the 6G continuum, network transformation, network control programmability and telemetry, 6G devices, radio protocols and sensing. The basic framework for the development of System PoCs and System-PoC #A was first described in Hexa-X-II deliverables D2.1 [HEX223-D21] and then D2.2 [HEX223-D22]. In this deliverable a detailed description of System-PoC #B is given, accompanied by various results.

In section 3.1, a summary of the three System-PoCs and their incremental -in terms of 6G feature availability- methodology is provided. Next, a mapping of the components constituting System-PoC #B to the 6G system blueprint is presented in section 3.2. In section 3.3, the System-PoC #B's architecture is described, followed by the components that comprise System-PoC #B. In section 3.4, the designed E2E scenarios that are implemented for the experimental evaluation are presented. In section 3.5, the related KPIs and KVIs to the System-PoC #B's evaluation results are analysed, along with System-PoC #B's results. Lastly, in section 3.6 simulation studies performed to support performance evaluations are presented to further complement the PoC based evaluations.

3.1 Overview of the E2E system evaluation and validation activities

The Hexa-X-II project aims to transition from the initial development phase of 6G technology to defining a well-integrated 6G system, prioritizing social, environmental and economic sustainability. This requires a comprehensive approach that includes novel devices, infrastructure, radio and network capabilities, E2E management & orchestration, as well as security and system-level resilience. To achieve this, three system-PoCs are planned to be developed incrementally throughout the duration of the project.

- The first iteration (System-PoC #A), whose detailed description and results are reported in D2.2 [HEX223-D22] and D2.3 [HEX224-D23], focused on smart network management aspects for demonstrating management mechanisms.
- The second iteration (System-PoC #B), which is presented in this deliverable along with its results, is designed to build upon System-PoC #A by incorporating new enablers (e.g., pervasive technologies, network functions) also by assuming that not all of the network and cloud domain resources, application components and devices are part of the same domain and/or geographical location. Therefore, in this current System-PoC iteration it is required to orchestrate all involved components in a coherent and synchronised way. The integration of the new enablers as well as the enhancement of the previous ones, allows to broaden the sustainability aspects already considered in System-PoC #A, e.g., to investigate the data exposure security (social sustainability), the dynamic replanning of available devices roles (environment sustainability), etc.
- The third iteration (System-PoC #C), will focus on radio and device aspects, while further evolving the 6G architecture design and smart network management introduced in the previous System-PoCs.

3.2 System PoC E2E architecture and alignment with the 6G system blueprint

The main elements showcased as part of System-PoC #B comprise devices, management & orchestration features, intent-driven resource allocation, advanced application features, as well as novel architectural enablers, such as the introduction of flexible topologies, as well as aspects beyond communication, such as the exposure of compute / communication resources, sensing/localisation capabilities, etc. A set of functional as well as performance aspects are validated and discussed. In Figure 3-1 a mapping of these components to the 6G system blueprint is illustrated to indicate the aspects of the system's blueprint, covered by the System-PoC at this iteration.

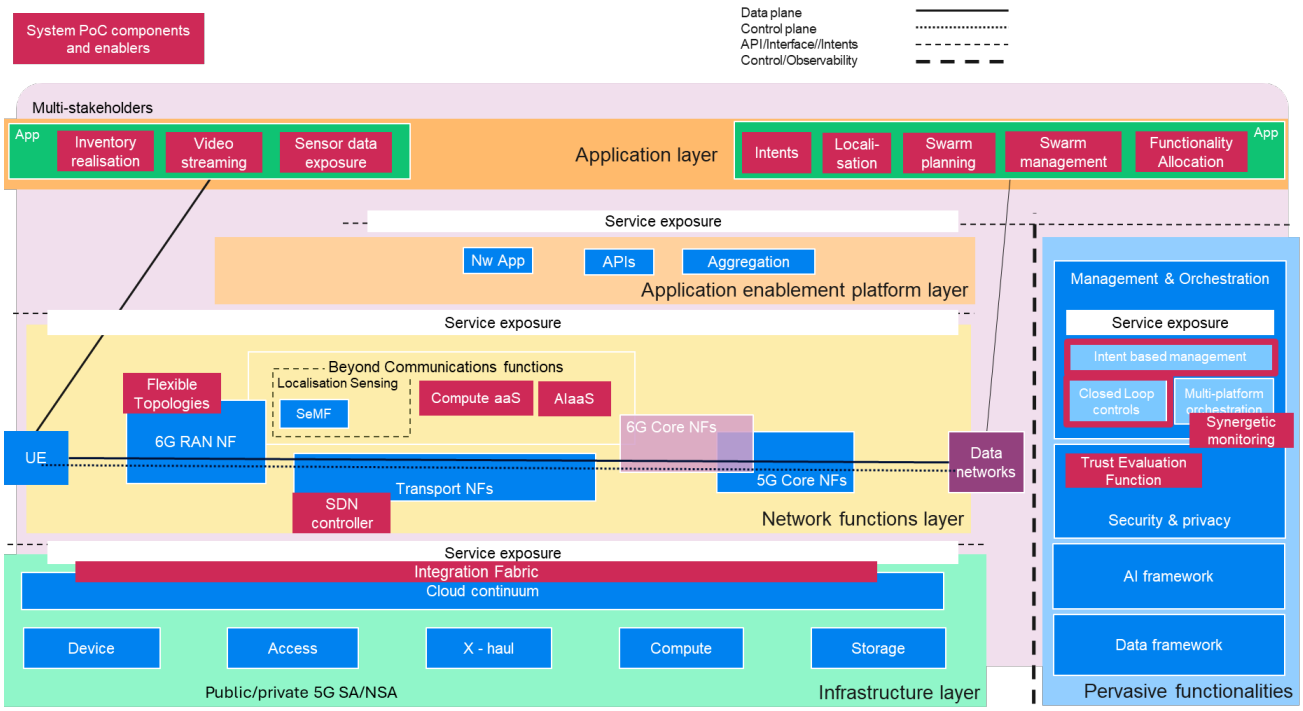

Figure 3-1: System-PoC #B components mapping to 6G system blueprint.

In the application layer of the system, all the System-PoC related applications, e.g., object detection, video streaming, latency-sensitive application, etc., are located. The network functions layer introduces software defined networking capabilities for the transport domain, while the flexible topologies enabler introduces novel capabilities in the PoC related to the exposure and dynamic exploitation of temporary communication and compute resources when required. Moreover, the compute as a service and AI as a service cloud (cf. sections 3.3.7 and 3.3.9), computing models are featured within the beyond communications functions. The integration fabric (cf. section 3.3.5) spans across the infrastructure layer, facilitating the seamless integration of the PoC's diverse systems and services. Additionally, within the infrastructure layer, the devices and compute resources are included. Finally, in the pervasive functionalities layer advanced management & orchestration aspects (cf. section 3.3.1), such as intent-based management (cf. section 3.3.2), closed loop controls (cf. section 3.3.4) and multi-platform orchestration (cf. sections 3.3.3 and 3.3.6), are included, along with the trust evaluation function (cf. section 3.3.8) featured within the security and privacy block. A detailed description of the components' implementation follows.

3.3 Components of System-PoC #B

A detailed description of the components of the E2E architecture follows, along with the System-PoC #B implementation scenarios.

3.3.1 Application and M&O features

Automated inventory management solution

As described in D2.2 [HEX223-D22] one of the configurations used for exhibiting the system's capabilities included a collaborative robot (cobot)-powered warehouse inventory management scenario. The key components include state-of-the-art fusion of computer vision and sensor data, to accurately identify, count and localise objects in real-time, along with a dynamic translation system between symbolic warehouse locations and 3D geometric coordinates. This enables seamless navigation for drones and autonomous mobile robots (AMRs) and accurate inventory pinpointing. More specifically, the following application components are described:

- Inventory realisation: This component encompasses all the specific functionality related to the identification and tracking of the warehouse items (including computer vision modules).

- Video streaming: The service that implements the transfer of the AMR- and unmanned aerial vehicle (UAV)-oriented data streams towards further processing.
- Sensor data exposure: The mechanisms and interfaces, which enable the AMR/UAV/infrastructure sensor information to be accessible by the respective data consumers.
- Intents: The user interface-based tool, enabling the warehouse manager/application end user to insert high-level requests/commands towards the system, related to the inventory operations' performance, duration, resource requirements, etc.
- Localisation: The service that encompasses all the required functionality and logic in order to provide location information to the respective consumer modules (e.g., swarm planning and management).
- Swarm planning/deployment: This service implements the functionality related to the reservation of the AMR/UAV resources to perform the inventory operations.
- Swarm management: The service responsible for the monitoring of the swarm during runtime, the identification and handling of events that arise (e.g., battery depletion, hardware malfunctions, etc.)
- Functionality allocation (FA) has been studied and results have been reported in D2.2 [HEX223-D22], D2.3 [HEX224-D23], where the objective was to optimize the placement of a) the inventory management; b) workloads requiring considerable computational resources.
- Trust Evaluation Function, which is responsible for analysing data (real-time and historical) coming from the compute nodes of the system and assessing the trust indices of each node. The trust indices are utilised by the FA component to place workloads on compute nodes with higher trust indices, aiming to maximise the system's overall trustworthiness.
- Synergetic Monitoring: Service enabling the exposure of infrastructure, device, and network monitoring metrics of multiple domains at a common location towards the joint optimisation of resources.

6G latency sensitive application

As described in [HEX224-D23], the application consists of two workflows, a streaming pipeline for live streaming the footage from the robot to the frontend and an object detection which is sampling the frames and detecting possible danger situations in the manufacturing facility.

The two workflows define the workloads that are going to stress the network and the computing infrastructure as follows (also displayed in Figure 3-2):

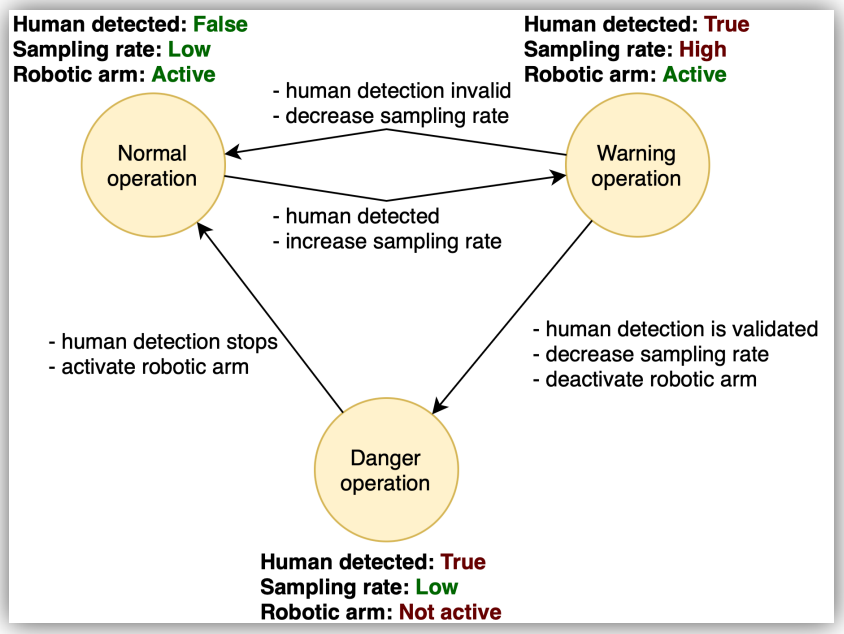

Figure 3-2: 6G service – object detection state machine.

1. Object detection – Arm control
    a. Automated loop
        i. Object detection through the robot surveillance camera is continuous with a low (L) sampling rate (normal operation).
        ii. Once a person is identified in the camera footage, the object detector tries to validate if there is indeed a human in the location by increasing the sampling rate (i.e., we move to warning operation).
        iii. If there is no human, we return to normal operation. If the module continues to identify a human in the incoming footage, an alert is sent to stop the robotic arm and the sampling rate is decreased (i.e., we move to danger operation).
        iv. Once the human is out of the camera's scope, the object detector reinitiates the arm's operation and we return to normal operation.
    b. Workload
        i. During the service's **normal operation,** the sampling rate is low, which means that the traffic workload is low (L).
        ii. While the detector tries to validate if there is a human in scope, the sampling rate is high (H), which introduces higher traffic in the service chain (**warning operation**).
        iii. If a human continues to be detected, the robotic arm stops, and the sampling rate is reduced again to low (L) **(danger operation).**
        iv. When a human is no longer detected, we go back to normal operation where the sampling rate and the traffic workload are low.
2. Live video streaming: This workflow is a high-bandwidth, high-availability scenario that runs continuously and offers a communication and computational challenge for maintaining its continuous operation.

A high-level schema of the application is shown in Figure 3-3, where the two workflows are illustrated, as well as the services' connection to the browser and the robots. The displayed stopwatches indicate the different monitoring points used for measuring partial and end-to-end latencies. Thus, the application workflow is traced from beginning to end counting four distinct time spans:

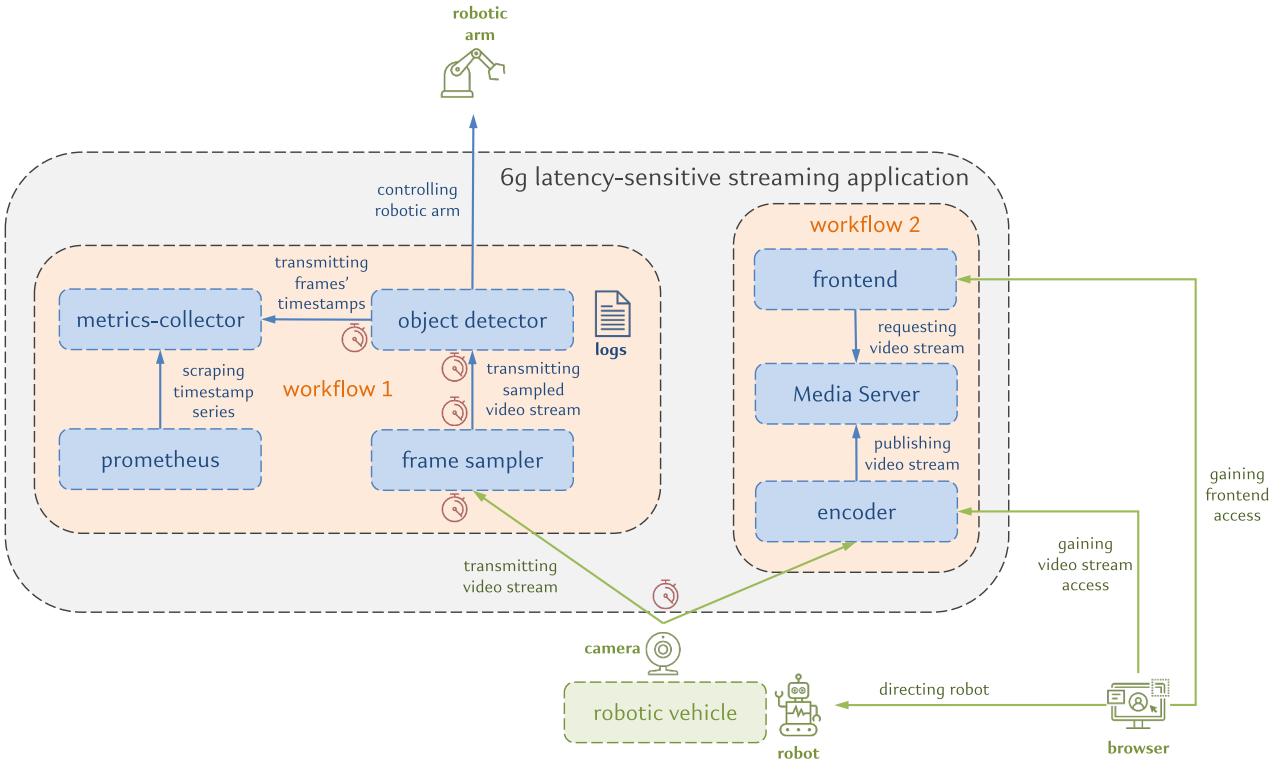

Figure 3-3: 6G application object detection and streaming workflows.

- Vehicle - Frame sampler
- Frame sampler computation
- Frame sampler - object detector
- Object detector computation.

Each time span corresponds to either a computation span or a communication span. The objective of the service, which is to minimize end-to-end latency, can be calculated as the sum of the four spans.

3.3.2 Intent-based Network solution

Intent-Based Network (IBN) represents a transformative evolution in network management, aiming to bridge the gap between business objectives and IT infrastructure. IBN belongs to the Intent-based Management (IBM) module within the 6G System blueprint (Figure 2-1), a module involving different kinds of intent-based requests. In the specific case of IBN, the focus is on network management requests. Traditional network management requires extensive manual configurations and constant oversight, which can be error-prone and time-consuming. In contrast, IBM allows network administrators to define high-level intents, such as prioritizing traffic for critical applications or enforcing security policies across all devices, although this relies on proper development and implementation of intent translation and provisioning functionality.

The IBN system is capable then of performing an automatic translation of these intents into network configurations by assembling different technologies like Natural Language Processing (NLP) and data science.

Within the System-PoC #B, the aspects regarding the IBM are managed using a solution based on some of the functions composing the Digital Service Manager-Intent Based Entity (DSM-IME), with the main reference being the enabler "Intent Translation and Provisioning" presented in [HEX223-D22, HEX224-D23]. The implemented solution called as "Intent-Based Network – Intent management Entity" (IBN-IME) and presented in Figure 3-4 and to demonstrate its closeness to its sources it is called as "Intent-Based Network – Intent management Entity" (IBN-IME). It is worth to mention that, compared to the enabler description previously presented in D2.2 [HEX223-D22] and D2.3 [HEX224-D23], the implementation of this enabler used in the System-PoC #B was evolved by adding a "Reporting" module as illustrated in Figure 3-4. By doing so, it was possible to have an IBN solution with the key elements to be used within the System-PoC #B phase.

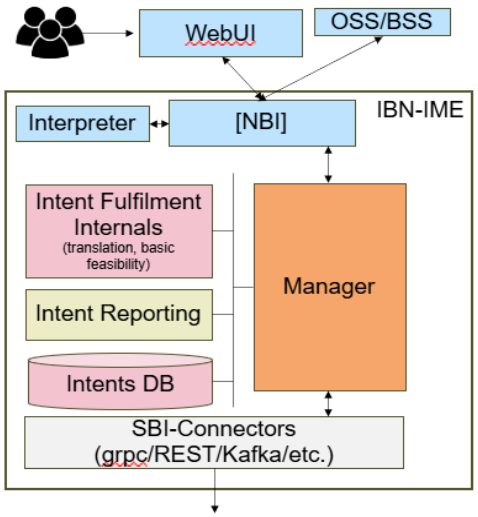

Figure 3-4: IBN-IME solution architecture.

Most of the modules are properly described in D2.3 [HEX224-D23], but in here we detail the use of the Web User Interface (WebUI) and the Reporting modules:

- **The WebUI:** The role of this module consists of permitting the communication between the user and the system through a web interface.
- **Intent reporting:** To inform about the status of the request, an object with a different structure is generated. Hence, the intent reporting component would have the role of generating the report and storing its information in the database.

In order to clarify how the presented architecture works, the workflow will be explained following an example. The example selected allows the user to perform the following request through the user interface: ***"create a cobot application on warehouse 24 with zero downtime"***.

The workflow (Figure 3-5) would start with the webUI interface triggering the corresponding operation which in this case would be "create intent requests". The webUI interface then forwards the information to the North-bound Interface (NBI) which acts as a filter identifying that for this case, a new intent request would need to be generated and hence some NLP would need to be triggered. Finally, the Interpreter would perform a keyword search with one-hot encoding searching to identify different types of information. As a main result the following information would be extracted out of the user request:

- **Identified operation:** create.
- **Identified resource:** cobot application.
- **Identified locations:** warehouse 24.
- **Identified recursivity:** True *(since it requested with zero downtime).*

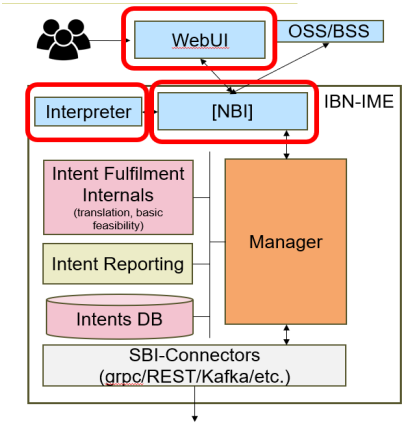

Figure 3-5: Workflow example: Intent reception and interpretation.

Then, the Interpreter would make the first blueprint of the intent request based on the included information by the user. The given information is then forwarded to the manager (Figure 3-6) which would trigger two main tasks. Firstly, it would perform a simplified feasibility check to verify that is viable to send the intent request. Secondly, it will also store the intent request information in a relational database using the data model illustrated in Figure 3-6.

Finally, a report is being generated and the South-bound Interface (SBI) carries on the work, by applying a filtering process to generate different types of requests. In this example, the SBI would establish communication with a service orchestration stack (more information in section 3.3.3). In addition, it ensures the necessary exchange of information is carried out.

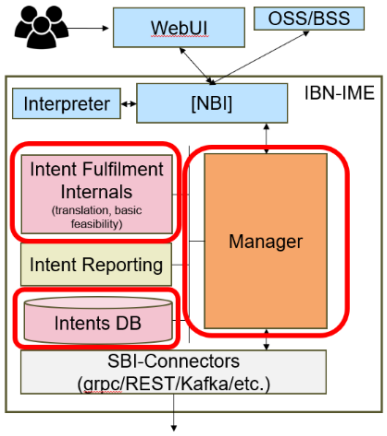

Figure 3-6: Workflow example: Intent blueprint creation, feasibility, translation and storage.

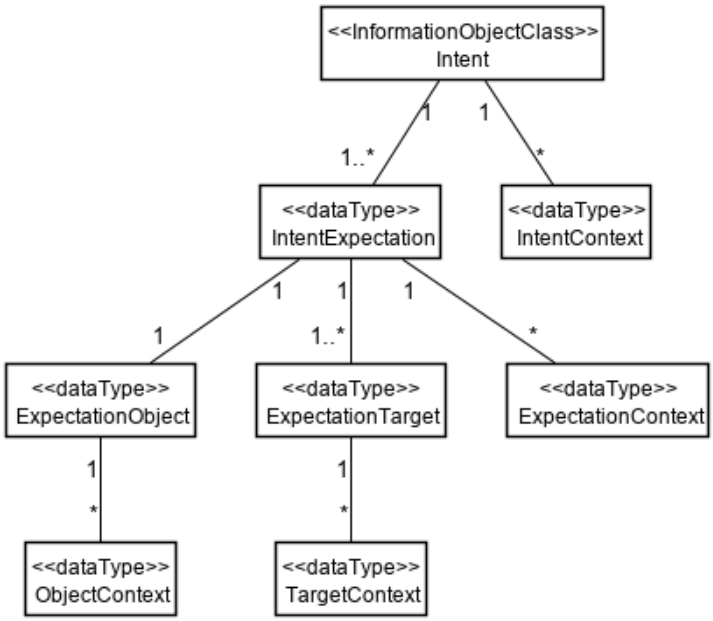

Figure 3-7: 3GPP Data model for intents used within the IBN-IME solution.

The current state of the IBN-IME is able to cope with user requests and process them from cradle to grave applied to a specific use case already demonstrated in [HEX2-AAG+23, HEX2-VAM+23]. In the current user requests, the NLP handles a type of resource referred as "cobot (collaborative robot) application" with an integration fabric that handles the service orchestrator requests in the SBI. Although the current state of the IBN-IME solution achieves the expected behavior, there are some features that will need to be improved or added. The following aspects will be studied, with the first two being near-future objectives (i.e., System-PoC #C):

- To replace the more traditional NLP capabilities with an autonomous agent which is able to automatize the wording recognition within the intent requests and to be able to automatically build a request without any middle steps.
- The integration fabric of the SBI would need to be adapted to cope with different resources and to trigger different kinds of vertical services like Kafka or google Remote Procedure Call (gRPC).
- Based on the improved NLP capabilities, to identify more quality aspects (e.g., maximum latency expected, minimum bandwidth desired, etc.) in order to update the Service Level Agreement (SLA) selection among the available set.
- The need to add an intent monitoring system to constantly validate its fulfilment and update the associated intent reports.
- The capability to manage intent conflicts with the functionalities to detect and resolve them.
- The use of close loops at the intent level in order to manage multiple intents and apply solutions when a monitoring system warns about them not being fulfilled.

3.3.3 (Service and) Resource orchestration

System-PoC #B demonstration includes the management of vertical services on a given set of cloud resources provided by different providers across the cloud-continuum. This is realized by the integration of two dedicated orchestrators for the services and the resources, respectively. The Service Orchestrator is central for the realization of the Closed-Loop (CL) automation (see section 3.3.4) although out of the Hexa-X-II scope. In order to sustain the realization of the PoC and fulfil the implementation and integration of the CL automation, System-PoC #B integrates a Service Orchestrator validated in other research projects [ESA23]. This orchestrator is considered as a blackbox component in the System-PoC #architecture.

The Resource Orchestrator (REC-EXEC) is an implementation of the *Multi-cluster resource manager*, one of the components belonging to the "Synergetic orchestration mechanisms for the computing continuum" enabler, more specifically the sub-enabler called "Multi-agent systems for multi-cluster orchestration" detailed in D6.3

[HEX224-D63]. In Figure 3-8 is depicted the high-level version of the REC-EXEC software architecture showing the main internal components.

- **Resource Manager.** Stores information related to the platforms and the devices, dynamically discovered through the Platform Manager. Such an information can be retrieved by 3rd parties (e.g., Service Orchestrator) through the REST NBI exposed by the Resource Manager.
- **Service Deployer.** In charge of application deployment (vertical applications, closed-loop functions, etc.) on the target cloud platform.
- **Service Repository.** Stores platform-specific orchestration templates e.g., Helm charts for K8s, retrieved by the Service Deployer at orchestration time.
- **Platform Manager.** Encompasses a set of technology-specific driver to interact with the different target cloud platform supported enabling the dynamic discovery and continuous monitoring of cloud resources in the continuum.

A more detailed version of the REC-EXEC architecture along with the defined operational workflows and information models is reported in D6.3 [HEX224-D63].

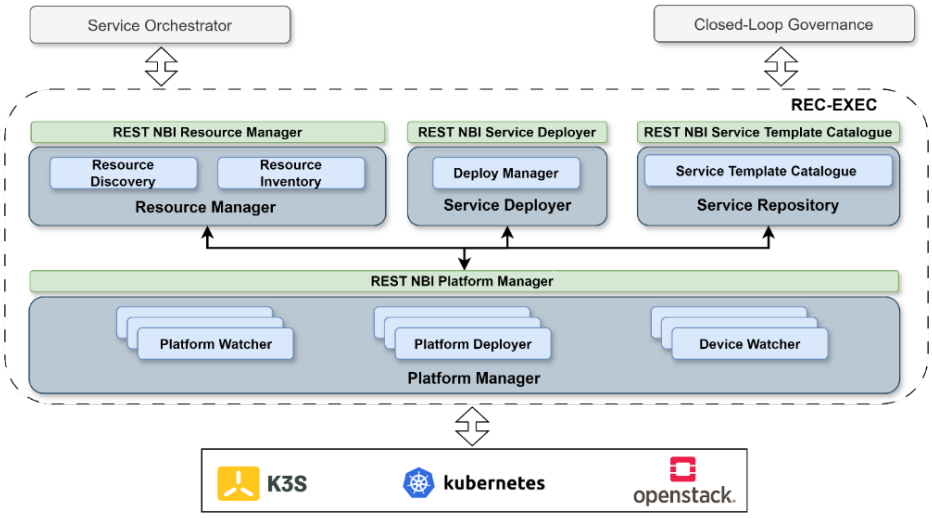

Figure 3-8: Resource Orchestrator high-level architecture.

3.3.3.1 Role in the PoC

The REC-EXEC has been integrated in the implementation of the demo scenario described in Section 3.4.2, (Cobot-based surveillance service management) where performs the following tasks:

1. Provision a Video Streaming application on a target cobot through the dedicated platform
2. Request the cobot platform to start the target cobot's patrolling
3. Provision the Closed-Loop functions at the edge cloud (a Kubernetes cluster) to enable service automation

Tasks 1 and 2 are requested by the Service Orchestrator as part of the service provisioning request while Task 3 is requested by the Closed-Loop Governance described in Section 3.3.4.

Figure 3-9 shows a piece of the REC-EXEC logs, where is shown the management of the request for Task 1.

```
2024-06-04T10:39:40.019Z : Completed initialization in 4 ms
2024-06-04T10:43:31.176Z : Requesting deployment of service video-streaming-app[video-streaming-app,] in cluster 3313d300-6e30-4e49-beff-fe002b3161
2024-06-04T10:43:31.178Z : Requesting deployment of component video-streaming-app
2024-06-04T10:43:31.178Z : Retrieving template info for template video-streaming-app with version a15db731-ffa1-45df-8a22-dd9609b65d70...
2024-06-04T10:43:32.498Z : Translating service component video-streaming-app in HELM deployment ...
2024-06-04T10:43:32.498Z : Retrieving template 805a8b90-3ba5-48c1-afbc-03e73e803dfe ...
2024-06-04T10:43:32.636Z : Chart template video-streaming-app stored at /tmp/tmp1430110083564053006
2024-06-04T10:43:32.637Z : Customizing template video-streaming-app ...
2024-06-04T10:43:32.679Z : chart video-streaming-app compiled
2024-06-04T10:43:33.029Z : Requesting deployment: {"type":"helm","platformConfig":{"type":"k8s","id":"27156329-0fe1-4e01-8208-eed9f85b79c1","config"
2024-06-04T10:43:34.160Z : Component video-streaming-app deployment successfully requested
```

Figure 3-9: REC-EXEC logs for video-streaming application deployment.

From the Kubernetes Dashboard (Figure 3-10) of the cluster at the extreme-edge, where each cobot is a node of the latter: the Video Streaming App is up and running in the cobot.

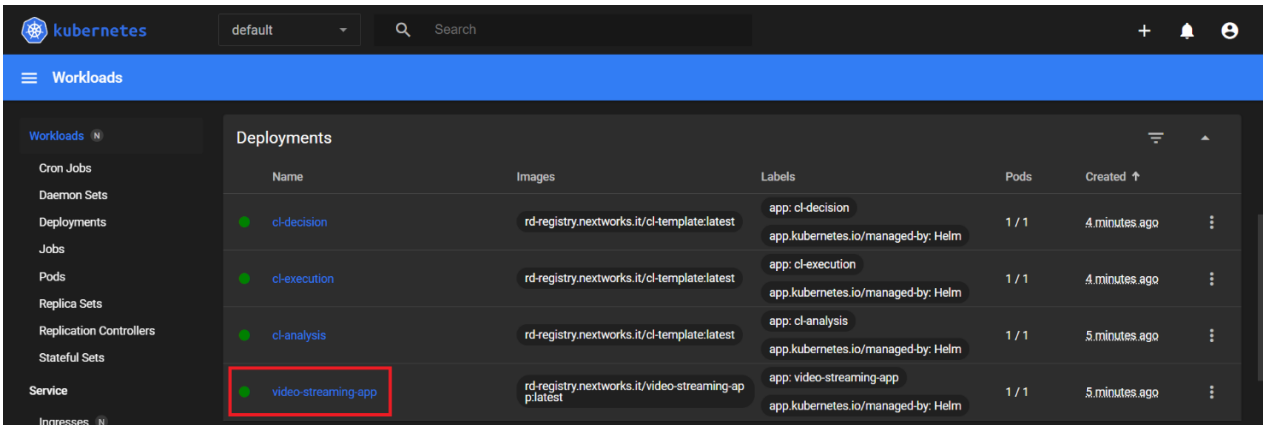

Figure 3-10: Video-streaming application deployed on target cobot displayed from edge Kubernetes Dashboard.

In Figure 3-11 the command sent to the Cobot Platform to trigger the patrolling the target cobot (Task 2) is presented.

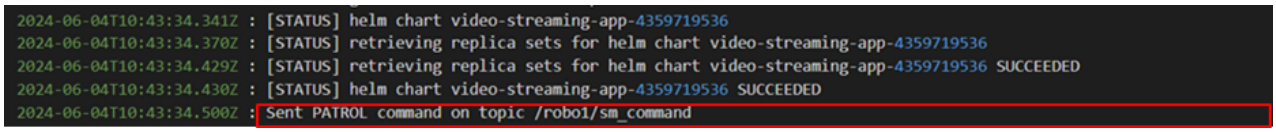

Figure 3-11: REC-EXEC logs for PATROL command sent to target cobot.

Similarly, for Task 3, Figure 3-12 and Figure 3-13 show the Closed-Loop deployment request management and the CL stages deployed on the K8s cluster at the Edge.

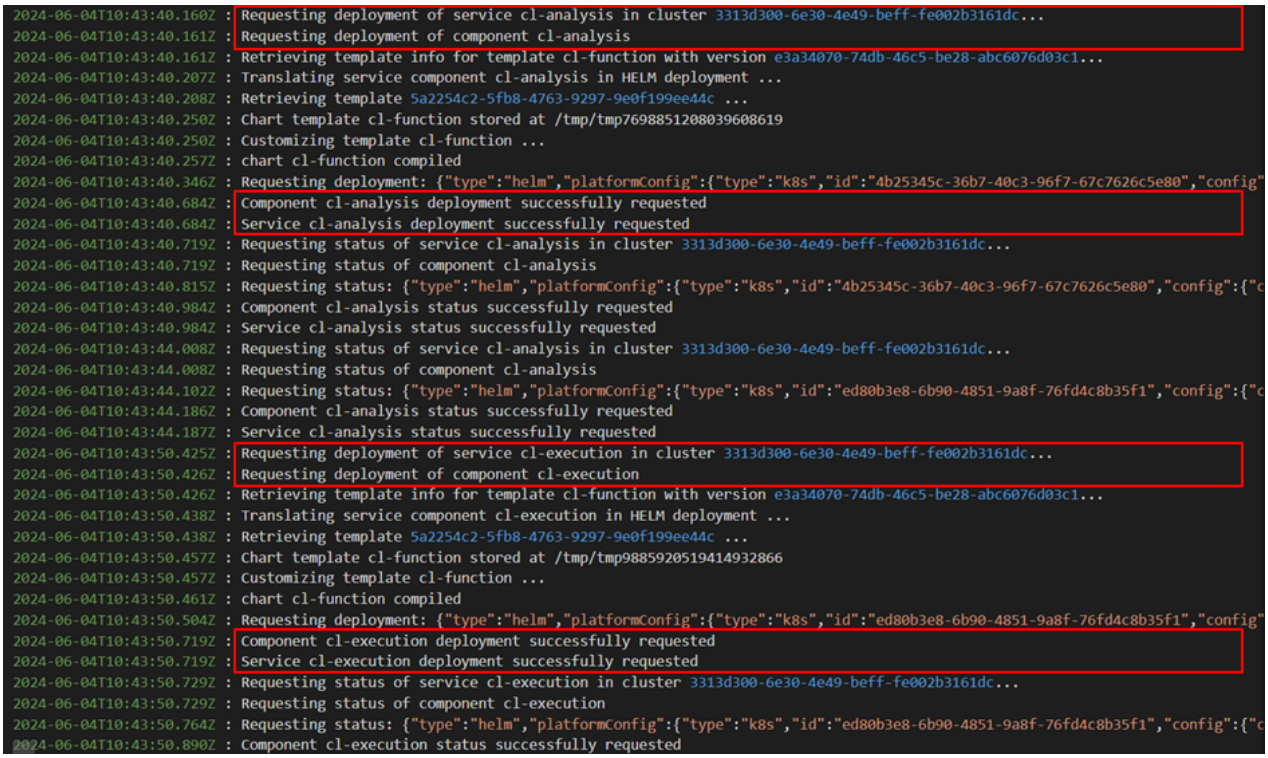

Figure 3-12: REC-EXEC logs for CL functions deployment, requests being sent to Closed-Loop Governance.

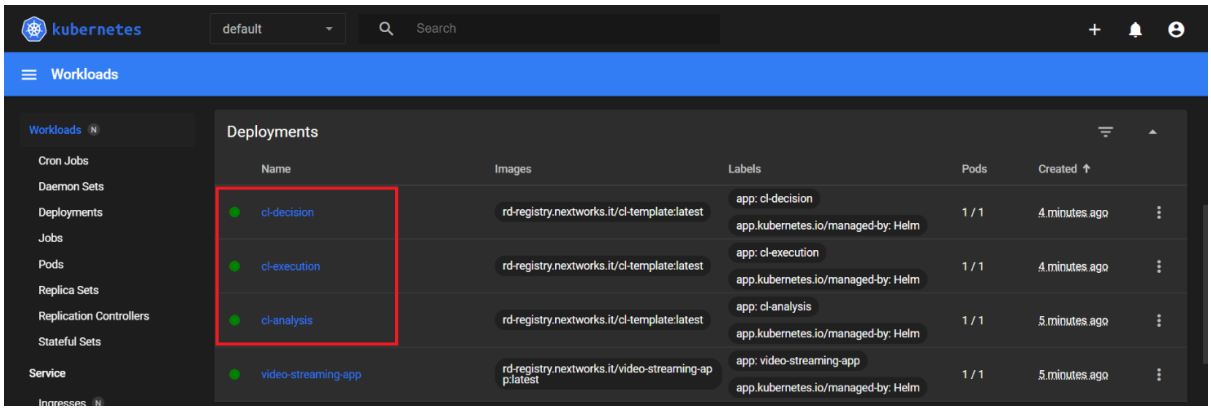

Figure 3-13: Deployed CL functions displayed from Kubernetes Dashboard.

3.3.4 Closed-loop automation

System-PoC B aims at providing a demonstration of Zero-touch automation based on the concept of Closed-Loop. The last evolution in the design of the closed-loops considers the different stages (monitor, analysis, decision, execution) completely orchestrable and configurable at runtime and provide specific functions for their management and coordination: multiple coexisting closed loops are indeed possible, making the automation truly pervasive in the 6G systems [HEX224-D63]. The functions for management and coordination of Closed-Loops, called CL Governance and CL Coordination respectively, are part of Enabler 8 *Real-time Zero-touch control loops automation and coordination system* whose last advancements in terms of design and implementation are reported in D6.3 [HEX224-D63]. The focus of this section is on the CL Governance since, at the time of writing, is the only CL management function demonstrated in the PoC. The integration of CL Coordination is in the roadmap for future enhancements of network and service automation of the PoC, as reported in D2.1 [HEX223-D21]. As per the REC-EXEC, a more detailed version of the Closed-Loop Governance architecture is reported in D6.3 [HEX224-D63], while Figure 3-14 shows its high-level version. It is important to note that the CL Governance manages the CL at runtime but is not directly responsible for its deployment which is in fact requested to the REC-EXEC.

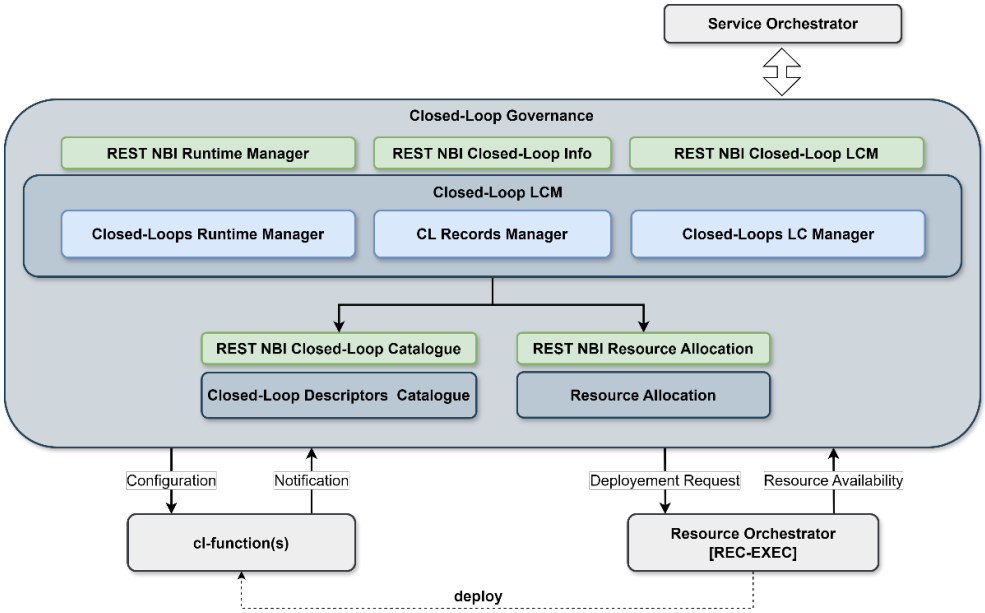

Figure 3-14: CL Governance high-level architecture.

- **Closed-Loop Descriptors Catalogue**. Repository storing CL and CL functions descriptors. It allows CRUD operations on the descriptors through its own REST NBI.

- **Resource Allocation.** In charge of computing CL functions' placement by exploiting information retrieved from the REC-EXEC: given the CL requirements, the Resource Allocation select the most suitable cloud platform among the available.
- **Closed-Loop LCM**. Is the core module of the CL-Governance, responsible for the lifecycle management of the Closed-Loops. It relies on 3 main sub-modules
    - **Closed-Loop LC Manager.** Selects proper CL and CL functions' descriptors given the service requirements, provisions and decommissions CLs
    - **CL Records Manager.** Tracks the existing instances of CL.
    - **Closed-Loop Runtime Manager.** Performs runtime operation of existing CL instances, e.g., configuration, start/stop, etc.

The CL Governance is part of the scenario described in Section 3.4.2 and is in charge of the management of a specific CL to enable the zero-touch automation for the cobot-based surveillance service.

The CL Governance requests the REC-EXEC to trigger the deployment of Analysis, Decision, and Execution stages of the CL, while the Monitoring stage is realised by means of a Programmable Monitoring Platform, still reported in D6.3, which is one of the implementations of the Enabler 2 *Monitoring and telemetry framework*. This stage is directly configured by the Service Orchestrator.

The stages are deployed as application at Edge K8s cluster and available through the CL Governance GUI, as shown in Figure 3-15, Figure 3-16 and Figure 3-17.

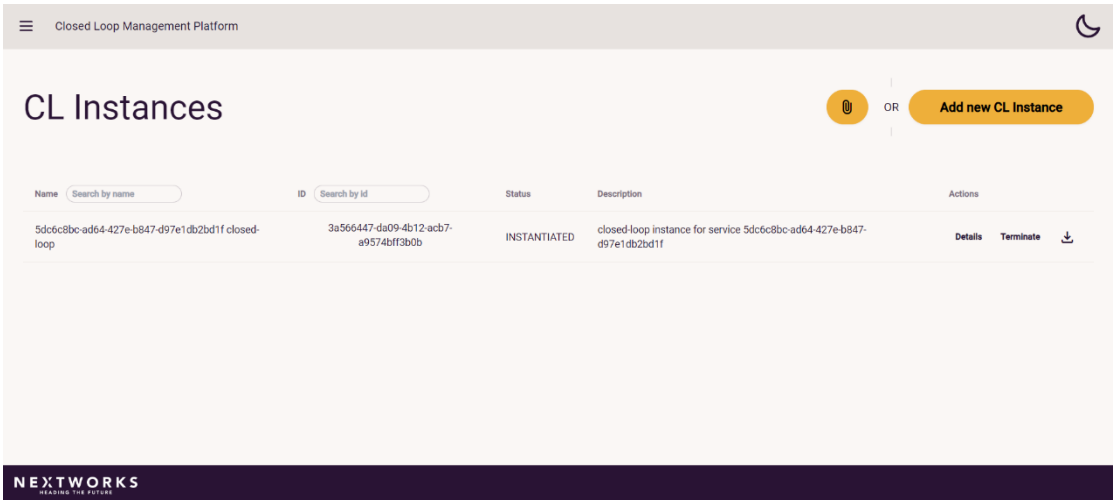

Figure 3-15: CL Governance GUI displaying the instantiated CL.

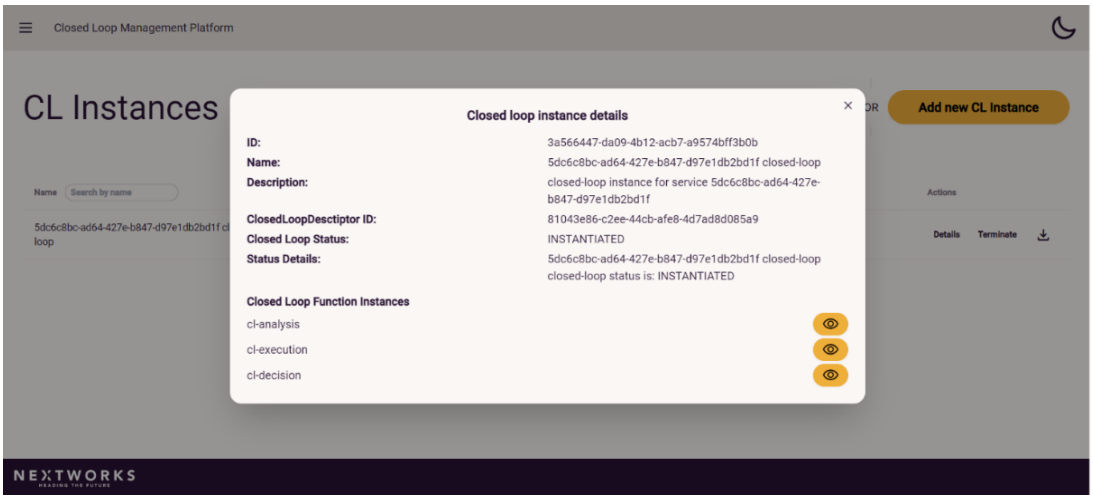

Figure 3-16: Details of the deployed CL instance displayed from the CL Governance GUI.

Figure 3-17: CL Analysis Function details and configurations displayed from CL Governance GUI.

The monitoring platform continuously monitors the status of the battery belonging to the cobot in charge of patrolling the area and when the charge drops below a given threshold (15%, in this case), the CL triggers the reconfiguration of the service through the interface of the Service Orchestrator, as shown in Figure 3-18.

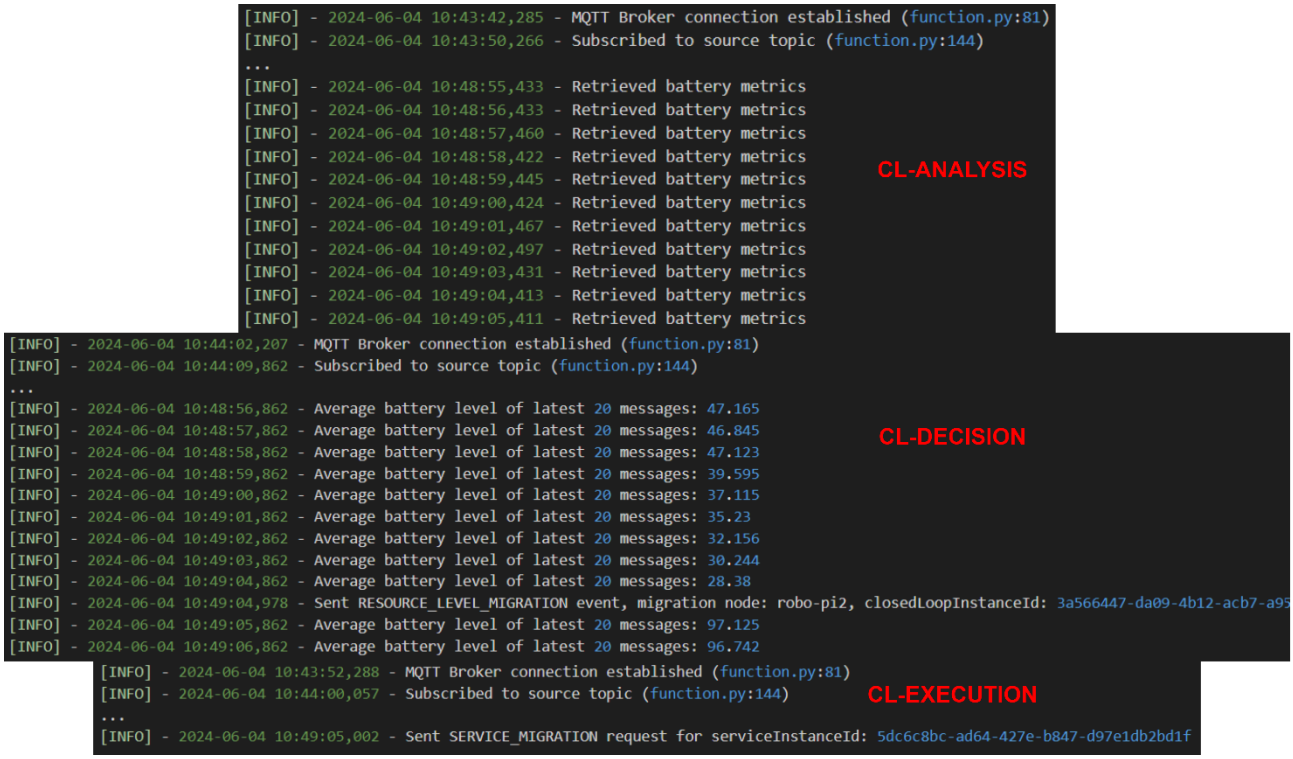

Figure 3-18: Logs of Analysis, Decision and Execution Functions.

3.3.5 Management capabilities exposure framework

The System-PoC #B is designed to demonstrate the role of the management capabilities exposure framework (MCEF) in providing a communication channel and exposing capabilities within the E2E architecture and to third-party entities. As previously presented in section 3.3 of D6.3 [HEX224-D63] within the scope of the “Integration Fabric”, the structure of the system is shown in the Figure 3-19.

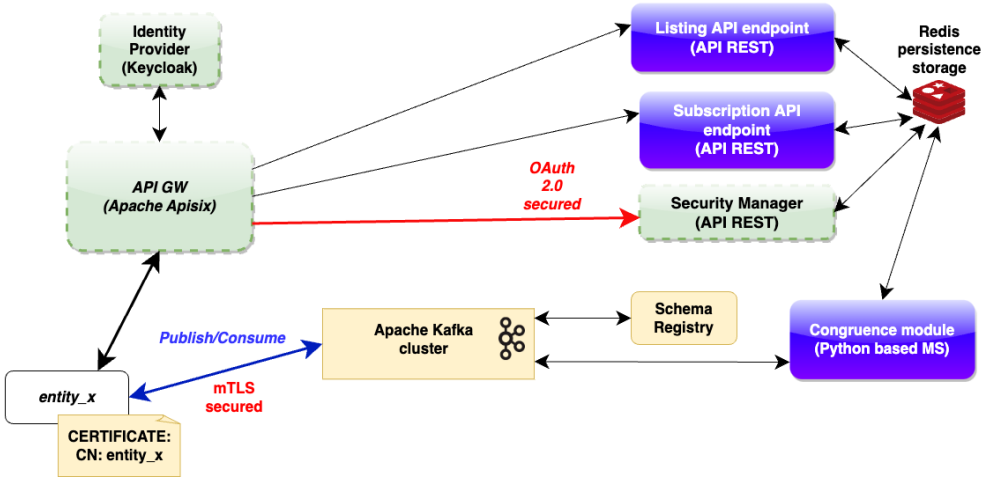

Figure 3-19: Management capabilities exposure framework structure overview.

The MCEF plans to offer interoperability within the scope of the PoC, leveraging the usage of an Apache Kafka cluster and several REST API endpoints. The Apache Kafka cluster, which is depicted in Figure 3-20, can be accessed via three exposed ports (9092, 9094 and 9096), each of which points directly to the broker of the data plane (i.e. the entity responsible for managing messages). Additionally, a REST endpoint is exposed. The aforementioned endpoint enables the direct production of data to the cluster via HTTP. This adds value by facilitating accessibility to entities which do not possess an Apache Kafka client. Communication with the cluster and within the cluster's nodes is safeguarded by means of mutual Transport Layer Security (mTLS). This security layer employs the use of ad hoc private keys and certificates for each entity. The credentials are delivered in the form of keystore files in a PKCS#12 (Public-Key Cryptography Standards) format. All entities that are integrated into the MCEF utilise these credentials in order to establish an encrypted connection and to set up authorisation at the cluster level using Access Control Lists (ACLs). In fact, each entity that is integrated is identified through the use of the Common Name specified in the generation of the keystore.

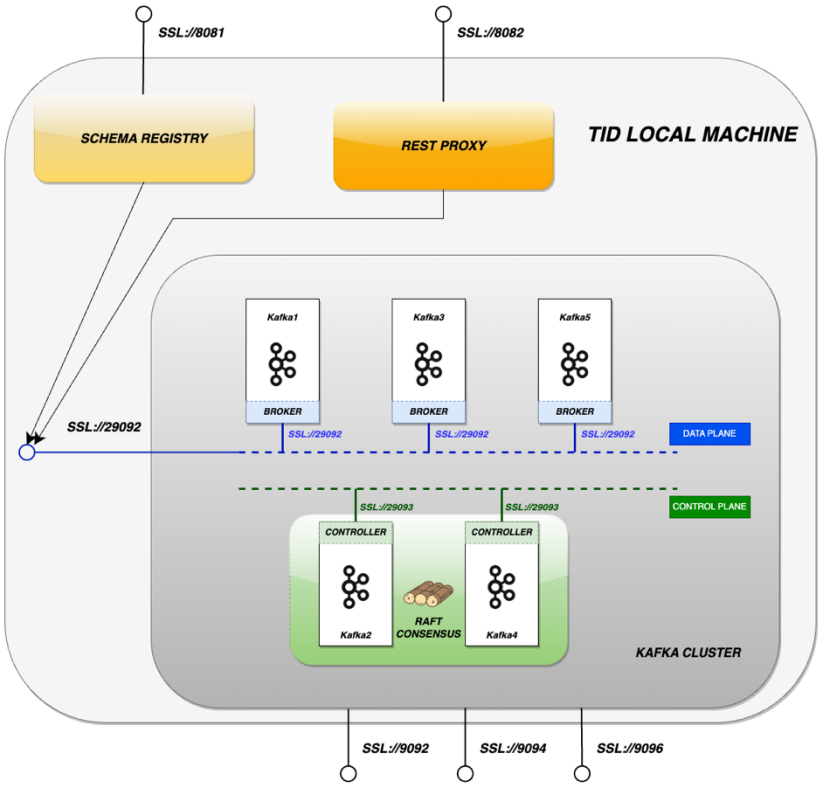

Figure 3-20: Management capabilities exposure framework's Apache Kafka Cluster.

The REST endpoint's structure overview is shown in Figure 3-21. Each of these endpoints is in compliance with the OpenAPI 3.0 API [OAPI24] specification, thereby providing standardised interfaces accessible via well-defined requests over HTTPS. As can be seen, there are three endpoints:

- Subscription API endpoint: this REST endpoint allows entities wishing to use the framework to perform onboarding and to create new communication channels, i.e. new topics. Furthermore, it enables the reverse operations, namely offboarding, in the case of an entity no longer wishing to utilise the MCEF and deleting the communication channels no longer required for communication with the system.
- Listing API endpoint: it allows the user to list all the communication channels present in the system, as well as to obtain information on a specific communication channel. Finally, it allows the entity to obtain a list of all entities currently booted into the system.
- Security API endpoint: it enables the retrieval of the credentials mentioned in the precious paragraph, namely the public certificate of the server and the unique keystore, which are then used to establish a connection in mTLS with the cluster.

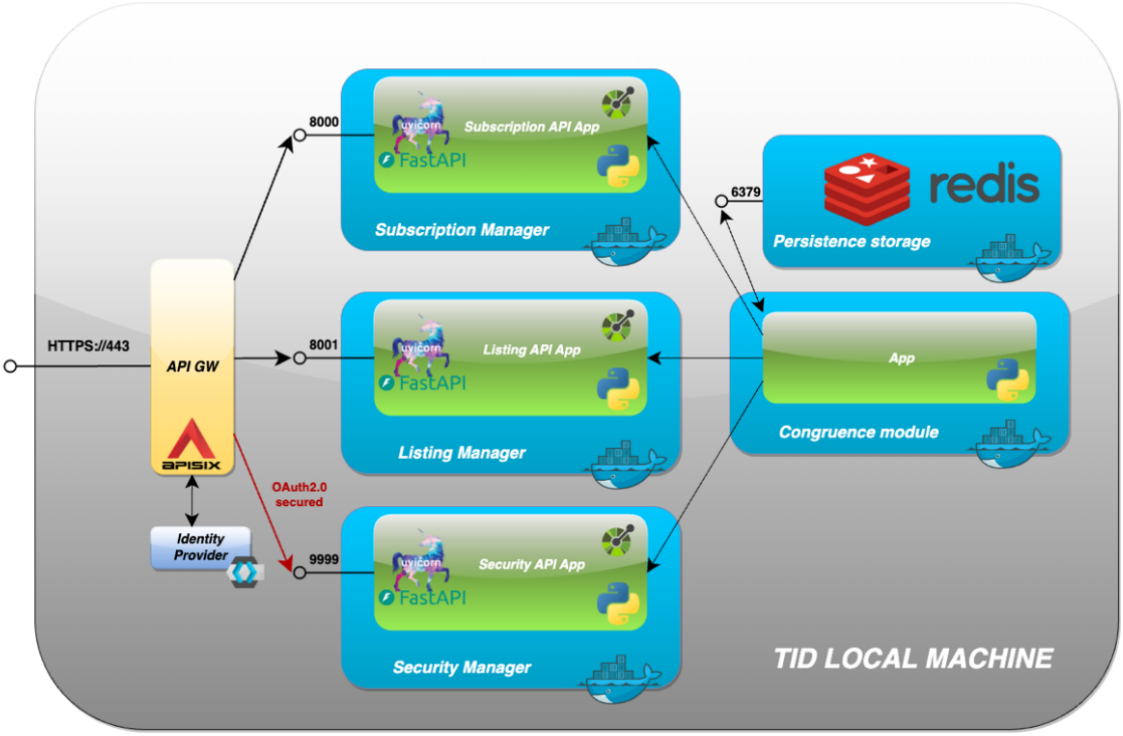

Figure 3-21: Management capabilities exposure framework REST API endpoints.

3.3.6 Programmable and flexible network configuration

This section presents the initial demonstration of Enabler Network programmability framework, which is addressed as SDN controller in Figure 3-1, and it is planned to be part of System-PoCs, in order to provide connectivity services upon a backhaul Transport Network. It delves into a "Transport DataPlane-in-a-box" approach, which demonstrates the capabilities of the TeraFlowSDN (TFS) controller to seamlessly integrate and manage both the packet and optical layers of transport networks. By employing gNMI (gRPC Network Management Interface) [GNMI] for configuring packet routers and close-to-real-time monitoring them through telemetry streaming, the controller ensures comprehensive visibility and control over network devices, thereby enhancing the overall efficiency and responsiveness of the network infrastructure. Furthermore, we discuss how the controller's use of OpenConfig [OPE24] data models and the ONF Transport API [ONF-TAPI] facilitates a unified and extensible framework for network automation.

The architecture of the TFS controller [TFS24], illustrated in Figure 3-22, showcases the ecosystem of micro-services that support the functionalities of the controller. Micro-services are small and loosely-coupled services with very focused and well-defined responsibilities, that cooperate with each other through networking-based communications through well-defined interfaces to realize the overall mission of the implemented software.

Micro-services also facilitate the deployment of the applications in cloud-native environments. Moreover, they enable the dynamic replication and scaling of the components of the software according to the traffic demands. This replicability capacity enables to replace failed replicas of the services easily and quickly, thus enhancing its resiliency and enabling self-healing of the software. Typically, these micro-services are software containerized, thus packing the actual software with its dependencies and requirements, such as software libraries. Thanks to this containerization, the micro-service—based architecture, and well-defined interfaces, each component can be implemented in different programming languages, thus increasing the modularity of the final product.

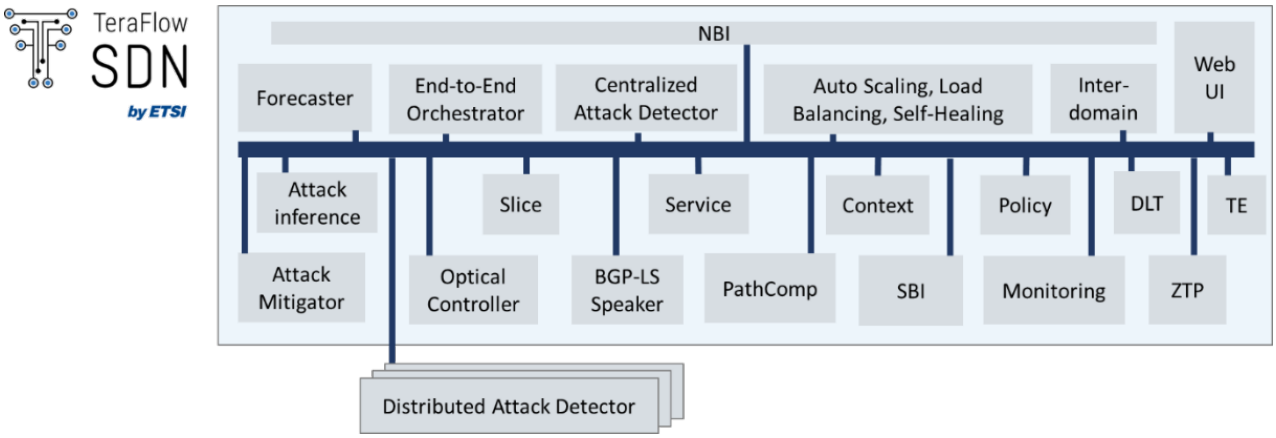

Figure 3-22: TeraFlowSDN Micro-service-based Architecture.

The core TeraFlowSDN micro-services that are relevant to this work include:

i. the Context component serves as a datastore for key informational entities, such as network topologies, devices (and their configurations rules), physical links, and connectivity services, among others. It is backed by CockroachDB, an open source and cloud-native NewSQL database supporting replicability, scalability, and high performance, while maintaining the classical relational database ACID (atomicity, consistency, isolation, durability) properties;
ii. the SBI is responsible for the management of the underlying network equipment both for configuring them and for collecting monitoring data from them. It features a pool of drivers out of the box, including NetConf/OpenConfig and gNMI/OpenConfig for packet routers/switches and optical equipment, P4 for disaggregated whiteboxes, and ONF Transport API for Open Line Systems, among others, and provides a Driver API and a plugin-based framework to facilitate adding support for new network elements;
iii. the Service component oversees the lifecycle of connectivity services within the network. It provides out-of-the-box a set of service handlers for different types of services, including optical connections, L2 and L3 VPNs, among others; it also offers a Service Handler API to facilitate adding support for new types of network connectivity services;
iv. the PathComp component is responsible for performing path computations. Whenever subservices, e.g. optical connections supporting overlay packet connections, are needed, it computes them and infers the appropriate configuration rules they require. The component features an Algorithm API to facilitate incorporation of new path computation algorithms and a plugin-based framework to onboarding them.
v. the Monitoring component enables to handle the collection of telemetry data from the network elements through the SBI and produce usable monitoring samples in the form of Key Performance Indicators (KPIs) tagged with their specific sample type, associated device and port, timestamp, and relevant measured values. This provides close-to-real-time and historical network data for human inspection, and for automated decision making through the Policy component (out of the scope in this work);
vi. the NBI serves as the communication bridge between TeraFlowSDN and external OSS/BSS systems. It exposes a set of interfaces that allow external systems to interact seamlessly with the controller, facilitating operations such as the exposition of the network topology and the provisioning and

management of connectivity services. It features a plugin-based framework to expose different delivery data models, mainly based on IETF standardized models; and

vii. the Web UI component provides a user-friendly interface for the network operators. The WebUI is integrated with Grafana to leverage its powerful filtering and visualization tools for plotting the monitoring data collected and processed by TeraFlowSDN.

Additional details on deploying and using TeraFlowSDN, as well as the currently supported SBI, NBI, and Service Types are provided in the ETSI GitLab Wiki pages [TFS24].

The workflow implemented in TeraFlowSDN to provision the packet+optical connectivity service is depicted in Figure 3-23. The workflow starts with the Operator requesting the creation of the Connectivity service through the WebUI (label 1 in Figure 3-23). This request passes through the WebUI and the NBI towards the Service component (2) that first stores the new service in the Context database with the status of "Pending" (3). Then, The Service component delegates on the PathComp the computation of the connections and sub-services (4). Whenever the path computation result is received from PathComp, the Service component first implements the underlying connectivity sub-services, e.g., the optical connections, through the SBI component sending the appropriate TAPI requests to the OLS controller (5-6-7). Then the Service component configures the overlay packet-layer connection and associated network instances on the routers again through the SBI component, but, in this case, using gNMI/OpenConfig-based requests (8-9-10). After completing the setup of the connections and activating them, the Service component updates the overall packet-optical connectivity service in the Context component database by setting its state to "Active" and stores the created sub-services and (sub-)connections (11). Finally, it returns the control to the WebUI and NBI (12) that reply to the Operator (13).

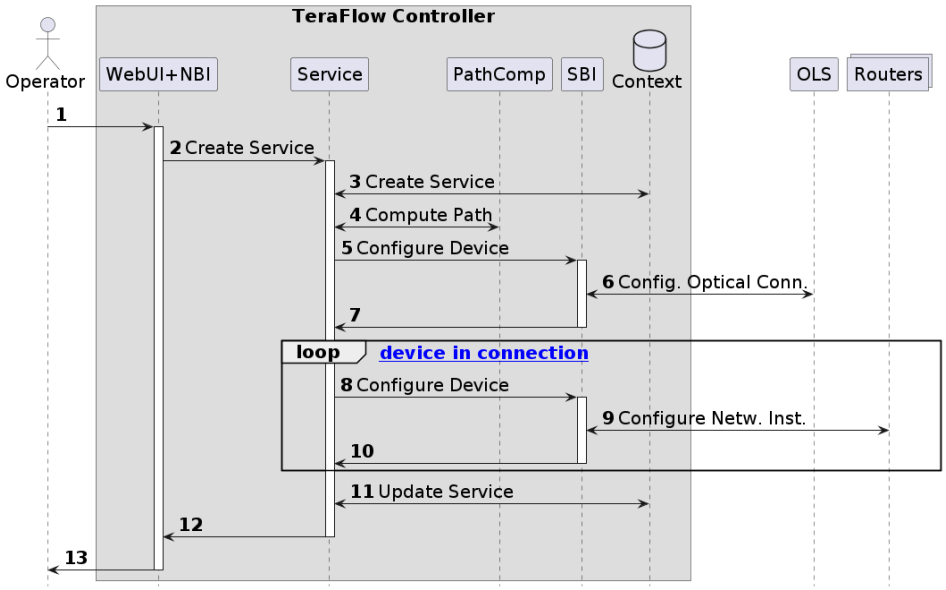

Figure 3-23: Create Service Workflow in TeraFlowSDN.

3.3.7 Training and inference of collaborative distributed machine learning model on a dynamically changing heterogeneous 6G architecture environment

This PoC enables collaborative training across application and core network allowing a joint Neural Network (NN) model to be trained for a video streaming quality estimation task. This component necessitates exposure of data in the form of NN model activations and gradient between application and network functions, and then training on the exposed data. Therefore, it is expected to have impact on the following enablers (red boxes) in Figure 3-1: AI-related functionalities in the 6G system blueprint such as video streaming, data exposure, and AIaaS. As presented in D2.3 [HEX224-D23] section 8.2.2, component PoC #B.2 is based on split learning, which is a distributed learning technique for vertical federated learning, and enables three main technical properties: i) enables cross-domain training and inference, without necessitating to move dataset between distributed data nodes; ii) enables model generalization that learns common representation of data to serve multi-tasks (use cases); iii) enables model layer offloading when computation and energy resources are scarce.

The distributed model architecture is illustrated in Figure 3-24. It consists of three input nodes: one receives physical layer dataset from gNB, another receives a medium access control (MAC) layer dataset from gNB, and the third one receives IP layer dataset from the user plane function (UPF) at the core network. Each three input nodes have local NN models. The output of the local NN models is sent to the NN model at the generalization node at the core network. The NN model generalizes the output (activations) that are received from the NN models at the input nodes for multiple tasks such as video playout bitrate estimation and video delay estimation. These two tasks are customized at the application at the output nodes. Observe that the dataset for the input features is located in the input nodes, and the dataset containing the ground-truth output labels are located at the output nodes. There are measurement points to monitor the performance, computation, communication, and energy overhead. The input nodes, generalization node, and the output nodes train vertical FL over iterative exchange of activations and gradients. More information and further details about the detailed signalling diagram of the PoC #B.2 are presented in [HEX224-D33].

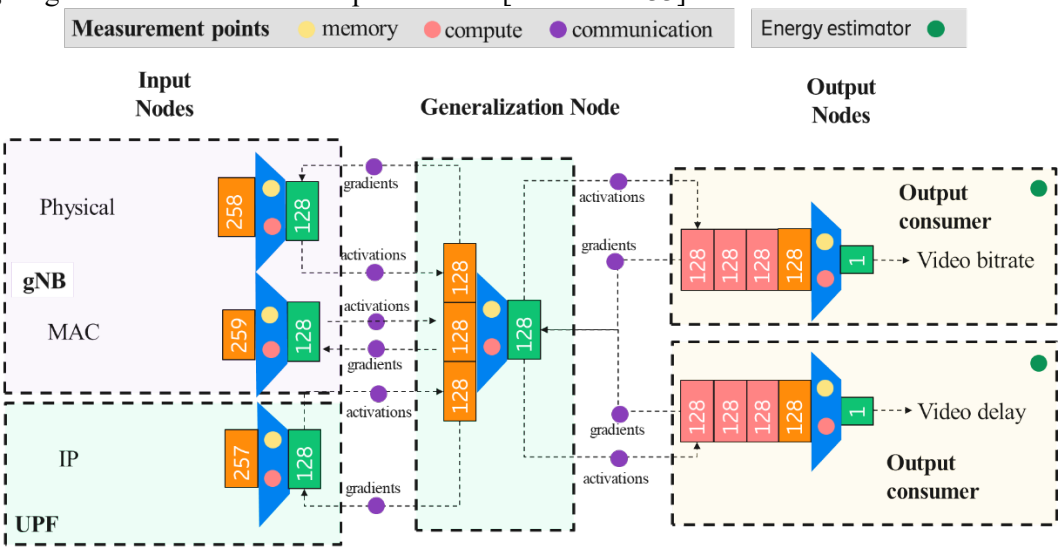

Figure 3-24: Distributed model architecture of component PoC #B.2 [HEX224-D33].

One of the prominent properties of the component PoC# B.2, which was earlier described in Deliverable 3.3 [HEX224-D33], is that it facilitates offloading NN model layers from a distributed node to adapt to the available compute, memory and energy. For example, an end-device running an application may employ the tail node of the NN model and the other part of the NN model may be deployed in the core network. This way, an application and the core network can jointly and collaboratively train an ML model via split learning. The main motivation of such a collaborative training, i.e., vertical federated learning, is that the collaborating parties then do not have to share local raw data in between. The dataset collected at the end device stays at the end-device, and the data collected at the core network stays at the core network. Via vertical federated learning enabled by the split learning technique, these parties can still benefit from each other's learning via the exchange of activations (so called smashed data) and gradients. The computation of activations and gradients need computation, memory, and energy as it consists of substantial amount of matrix multiplication operations.

The end-device may run on a limited battery, and a training or inference request may occur when the remaining battery at a device is low. In that case, depending on the criticality of the requested task, the end device may request to offload temporarily some of its NN model layers to the core network for training or inference. The illustration of the signaling diagram between the network data analytics function (NWDAF) and the application during model layer offloading is given in Figure 3-25. On the left, an NN model is deployed on the generalization node, e.g., at NWDAF, in the core network with K layers. On the right, there are two neural network models at the output nodes that are specialized for two different tasks: delay prediction and throughput prediction, where each NN model has L layers. In steps 1 and 2, the result of the NN model at the generalization node are sent to the application output nodes and is fed as input to two NN models (delay and throughput prediction tail nodes). The tail nodes then predict the delay and throughput. Observe that the interfacing NN model layer, so called cut-layer, has two neurons for alignment. In step 3, the energy estimation takes place

based on the NN model properties. The application may decide to offload some of the layers to the core network to reduce the energy consumed by the NN models. Steps 6 and 7 depict the request for model layer offloading from application to the core network. It appends the requested number of layers to offload, the layers that are intended to offload to the core network, and gradients. If an agreement is reached, the model layers that are intended to be offloaded are detached from the NN models at the application in Step 8. Backward propagation on the received gradients continues in the generalization node in Step 9. The incoming layers need to be aggregated in step 10 and then in step 11 attached to the generalization NN model at the core network. The training then can continue in steps 12 and 13.

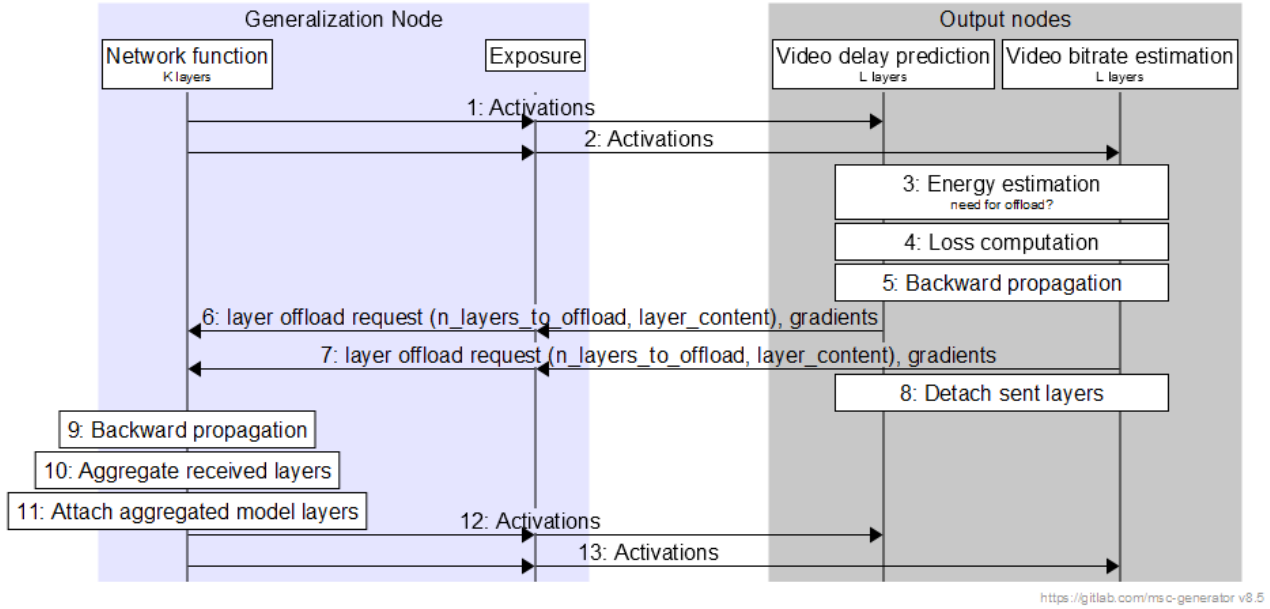

Figure 3-25: Illustration of tasks and signaling for model layer offloading from Application Output Node to the Network Function in the Core Network.

It is important to note that the component PoC facilitates offloading from generalization node to the output nodes similarly. In order for the application to be able to trigger a model layer offloading, it needs to estimate the energy consumption of the model training and inference accurately. To do that, we first performed experiments via Kepler tool that provides energy consumption values in real time while a ML task is running in a Kubernetes cloud. The energy consumption values are then mapped to the hyper-parameters and ML model training settings. Since the dataset is multi-dimensional, we trained an ML based energy estimator model, and compared it against the non-ML based estimator. The ML based XGBoost model was observed to be the most accurate estimator hence we deployed that in the output nodes in the Kubernetes pods that were emulating the application. In every round of training, the application estimates the energy consumption via the energy estimator model based on the current state of the ML model described by the model parameter size, estimated number of required rounds of training until convergence, number of layers in the NN model, number of neurons in intermediate layer, number of input attributes to the model as well as the batch size, i.e., number of samples fed into the NN model at every round of training. Please refer to Deliverable 3.4 [HEX224-D34] and 3.5 [HEX225-D35] for more detailed information on the component PoC #B.2.

3.3.8 Trustworthy flexible topologies and beyond communication aspects

Flexible topologies are designed to adapt quickly to resource and coverage constraints, making them essential for addressing the dynamic needs of 6G environments. These networks utilize components such as node discovery, trust/cost evaluation, and AI/ML-driven resource optimization to form adaptable and efficient network structures [HEX224-D33]. By integrating these elements, flexible topologies ensure robust communication and computation capabilities in various scenarios. An application regarding the proof of concept for flexible topologies is demonstrated in the warehouse inventory management scenario where a

Flexible Topology Node (FTN), implemented on an autonomous UAV with multi-connectivity capabilities, plays a crucial role in this setup. The FTN communicates with an edge server via a 5G network interface to maintain up-to-date information on Worker Nodes' (WNs) locations and tasks. As part of the Component-PoC#B.3 and System-PoC B activities, two scenarios trigger dynamic network allocation using the FTN: local connectivity loss and out-of-coverage intent. In both cases, WNs pause operations safely, with UAVs docking to the nearest AMRs. The WNs then enter a "network discovery" state to reconnect using the FTN's credentials. Once a reliable connection is established, the flexible topology component allocates the UAV to the optimal location. Key aspects of this demonstration include advanced M&O and flexible and trustworthy topology control mechanisms, as depicted in Figure 3-26, that interact with the advanced connectivity ecosystem and the autonomous operational units, ensuring state and policy feedback loops that enhance network performance and operational efficiency.

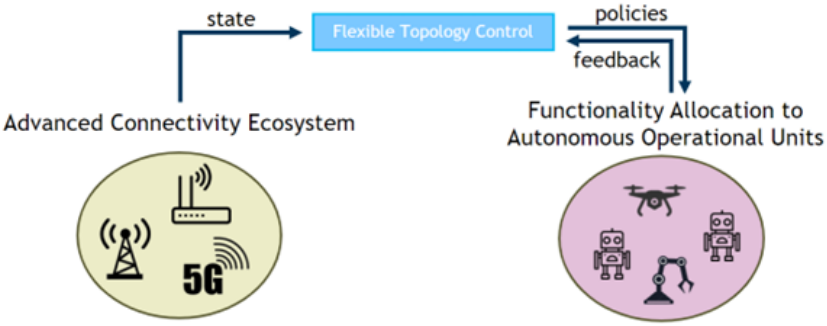

Figure 3-26: Advanced connectivity ecosystem and autonomous operational units.

3.3.9 AI-assisted E2E lifecycle management of a 6G latency-sensitive service across the compute continuum

Within the context of Component-PoC#B.1 [HEX224-D23], a developed microservices-based application revolves around smart manufacturing, explicitly focusing on real-time monitoring of manufacturing processes and ensuring safety through prompt responses to detected anomalies. The use case involves a stationary robotic arm performing specific functions whose operation must be automated. Additionally, a remotely controlled robotic vehicle allows a user to monitor the location. Figure 3-27 depicts the application graph representation of the developed 6G latency-sensitive service.

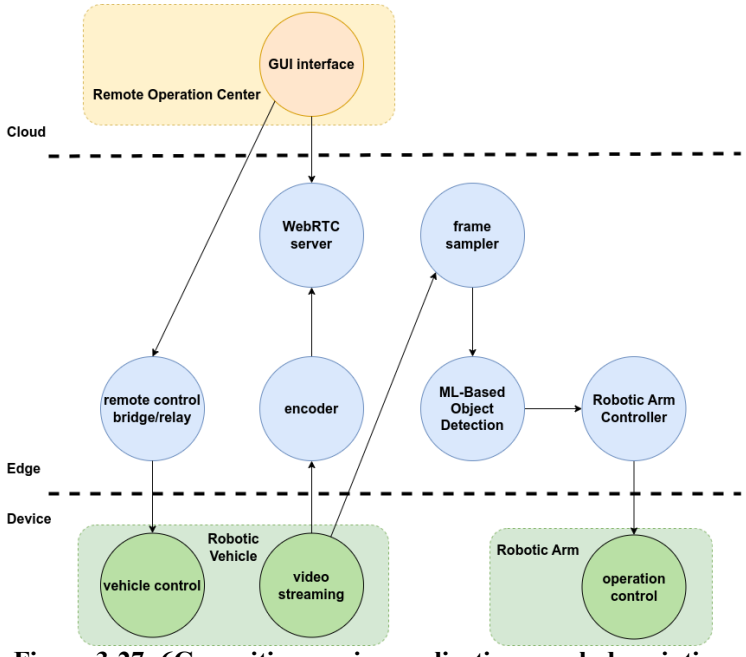

Figure 3-27: 6G-sensitive service application graph description.

Service placement occurs across three deployment layers: device, edge, and cloud. Overall, the provided functionality consists of two separate workflows: one related to the remote operation (i.e., tele-operation) of the smaller vehicle and another regarding object detection and alert generation, which control the robotic arm's operation. These two workflows are leveraged to showcase and evaluate the operation of the orchestration mechanisms and functionalities described in the next paragraphs and can be mapped to the 'Multi-platform orchestration' box in Figure 3-1.

3.3.9.1 Service autoscaling and migration

The object detection workflow is orchestrated using a set of autoscaling and computation offloading mechanisms across two clusters. Specifically, the ML-based object detector component is selected to be horizontally scaled as required based on the measured workload and end-to-end latency (at the application layer), being migrated from the edge to the cloud and vice-versa. A centralised joint autoscaling-placement mechanism is used as the intelligence agent managing the application. In specific, a reinforcement learning (RL)-based algorithm was designed to make the application adaptive to the variable incoming workload. These orchestration actions are decided based on the evaluation of real-time data regarding delays (reflecting computation and communication latency) collected at specific monitoring points. The autoscaling and placement mechanisms evaluate the collected data, and for every interval, the RL agent selects where the optimal placement is (taking into consideration resource and migration constraints) and how many replicas are needed for guaranteeing Service Level Objectives (SLOs) satisfaction in terms of end-to-end latency requirements. Further information about the implementation of the service autoscaling mechanism is made available in [Hex224-D63].

3.3.9.2 DLT-based service federation

Focusing again on the object detection workflow, we consider a two-cluster deployment but this time in a multi-domain scenario (i.e., a consumer and a provider domain, see Figure 3-28). When more resources are needed, a federation request is sent to initiate the federated deployment. Specifically, when increased workload and latency is noted, distributed ledger technology (DLT) federation is triggered by the consumer domain. Both provider and consumer domains agree on the federation terms with a blockchain smart contract (representing a dynamic SLA), and the object detector component is migrated to the provider domain's Kubernetes cluster. Once the deployment is successfully completed, the provider informs the consumer and shares relevant service information, including connection details. Finally, the consumer terminates its object detector component and reroutes the incoming video stream to the newly deployed federated object detector.

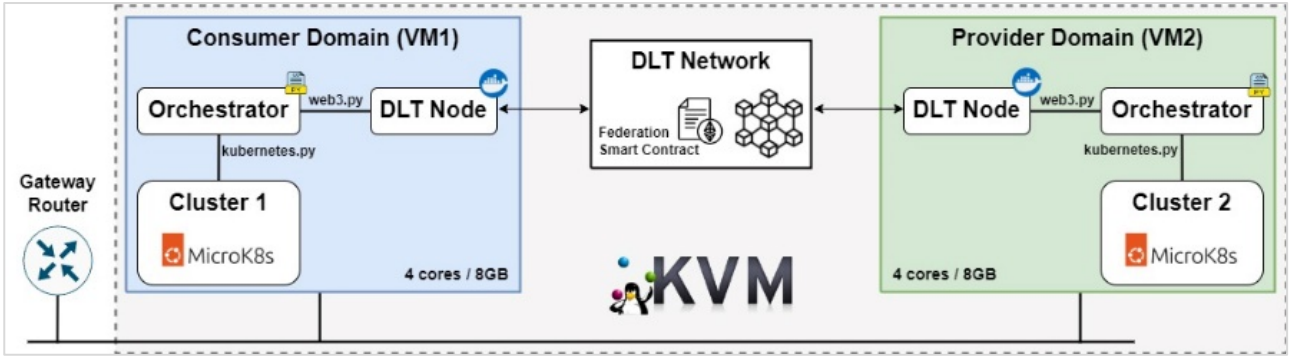

Figure 3-28: DLT-based service federation.

3.3.9.3 Extreme-edge cluster emulation on high availability scenarios

The video streaming workflow is used to demonstrate a high availability scenario on top of an emulated extreme-edge infrastructure. To that end, the Infrastructure Layer Emulator (ILE) is used to allow the deployment of large number of different computing nodes, different stakeholders and network domains as core, edge and extreme edge. The extreme-edge nodes are configured to be highly volatile with devices unexpectedly connecting/disconnecting. As the extreme-edge nodes turn off, the service is still running since more replicas are deployed in the remaining extreme-edge nodes. At the same time, the orchestrator redeploys the missing replicas in other powered-on extreme edge nodes, as required. An example snapshot of the ILE is shown in Figure 3-29. Further information about the ILE solution is available in [Hex224-D63].

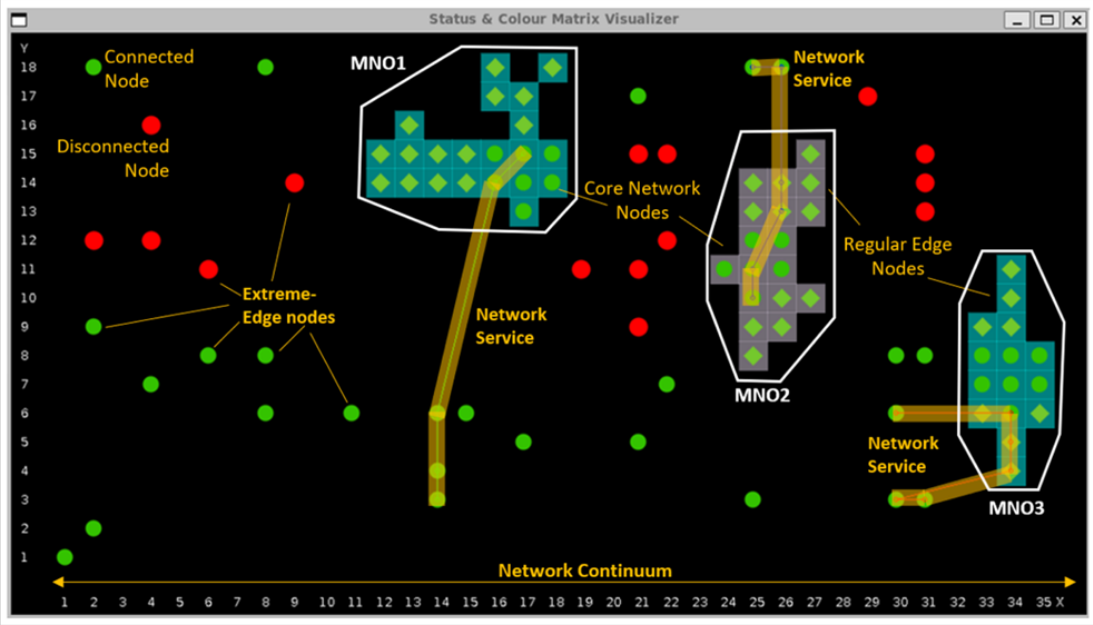

Figure 3-29: ILE high-availability scenario.

3.3.9.4 Inter-cluster communication scenarios

Two alternative options are explored regarding inter-cluster communication. First, the karmada multi-cluster management tool, and second, Network Service Mesh (NSM). The former is a management system that enables deployments across multiple Kubernetes clusters and clouds, leveraging Kubernetes-native APIs and providing advanced scheduling capabilities. The latter is a Hybrid/Multi-cloud IP Service Mesh that allows connecting workloads that run in multiple clusters by working in parallel with the existing container network interfaces (CNIs) and by requiring no changes to the workloads themselves, except for adding simple annotations to the underlying pods.

3.4 E2E Implementation scenarios

3.4.1 Autonomous operation

The implementation scenario of the automated inventory management solution addresses various enablers across different layers, including resource allocation workflows, in the context of which, various components interact to ensure optimal placement and execution of the scenario's computational tasks, such as the object detection workloads related to the inventory audit operation. In this workflow, an infrastructure monitoring component continuously monitors various KPIs and the status of physical and virtual resources within the system. If any metric exceeds a predefined threshold or if a device becomes unavailable, the infrastructure monitoring component triggers an alert to the API server. The API server also receives intent-based requests, i.e., high-level commands, from end-users or external sources. These requests express specific needs or objectives to be accomplished by the system. Upon receiving a request from either the infrastructure monitoring component or an external intent-based request, the API server triggers the functionality allocation mechanism, which is responsible for determining the optimal allocation of resources to meet the request's requirements. The functionality allocation mechanism needs detailed information related to

- the compute nodes capabilities and current status from the infrastructure monitoring,
- the computational workloads and tasks requirements from the service registry, and
- the compute nodes' trust indices, which are provided by the Trust Evaluation Function component

to make an informed decision. The API server then acts as an intermediary, gathering the requested information from the infrastructure monitoring component for resource status and capabilities; the service registry for task requirements and workload information; and the trust evaluation function for trust indices. The functionality allocation mechanism then performs an optimisation process using the gathered data. Based on the optimisation

process, a proposed placement solution is generated and sent to the API server. The API server verifies the feasibility of the solution and then forwards the decided action to the *orchestration enforcer*, which implements the placement decision on the system's resources. Once the orchestration enforcer completes the placement, it sends a confirmation response back to the API server. The API server then updates the infrastructure monitoring component and the service registry with the new placement information, ensuring the system's state is current and reflective of the latest changes.

3.4.2 Zero-touch Cobot-based video surveillance

The Zero-touch Cobot-based video surveillance is a demonstration scenario belonging to the System-PoC #B, involving several components of the PoC architecture, as shown in Figure 3-30:

- IBN-IME consumes and manages intents from vertical users. See section 3.3.2
- Management and Orchestration, for Zero-touch service and resource orchestration. See Section 3.3.3.
- Cobot's Facility, Cobots (extreme-edge devices), edge-cloud governed by a Cobot management platform. The Cobot Management Platform consists on a Kubernetes Cluster, with server and cobot nodes, and a Cobot Service Manager. While the former schedules the video surveillance services on cobot Kubernetes nodes and the latter consists on a series of nested State Machines (SM) which handle cobot operations through a Robot Operating System (ROS) MQTT bridge. In this implementation each cobot has its own state machine with three states PATROL, GO CHARGE and IDLE states. This implementation facilitates the service lifecycle management from the CL entities, which only uses two interfaces a MQTT Broker and the Kubernetes API for interacting with the Closed Loop enabler. The former for collecting metrics (battery) and interacting with ROS SMs, and the latter for scheduling services on the cobot nodes.

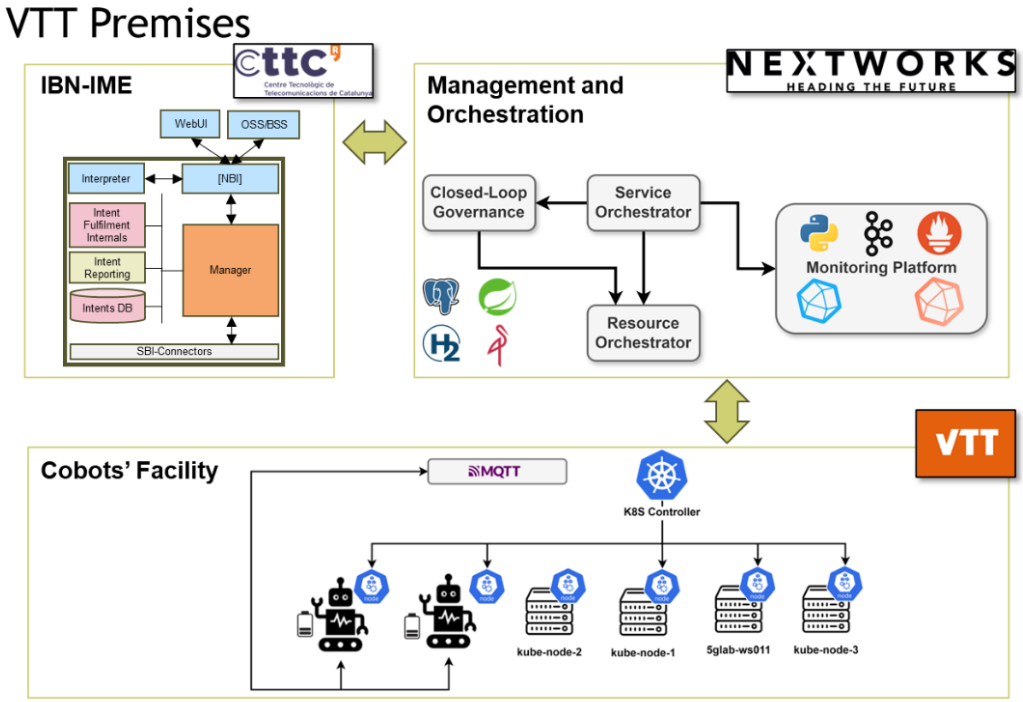

Figure 3-30: Zero-touch Cobot-based video surveillance scenario.

The scenario considers a vertical aiming at requesting the provision of a surveillance service based on patrolling cobots. The request is submitted in form of an intent, digested by the system and enforced on one of the available cobots: once the battery of the patrolling cobot falls below a certain threshold, a proper CL request the service migration towards another suitable cobot, guaranteeing the service continuity. The workflow is depicted in Figure 3-31 and Figure 3-32:

- A Vertical User request a cobot-based surveillance service by specifying an intent on the IBN Platform
- The IBN Platform parses and translates the intent in requests for the Orchestration Stack

- The Orchestration Stack provisions the service on the cobot's Management Platform. Additionally, it requests the Cobot Service Manager SM to change cobot1s state to patrol.
- In parallel specific Closed-Loop (CL) functions are deployed to guarantee the service continuity in case of cobot's battery drain.

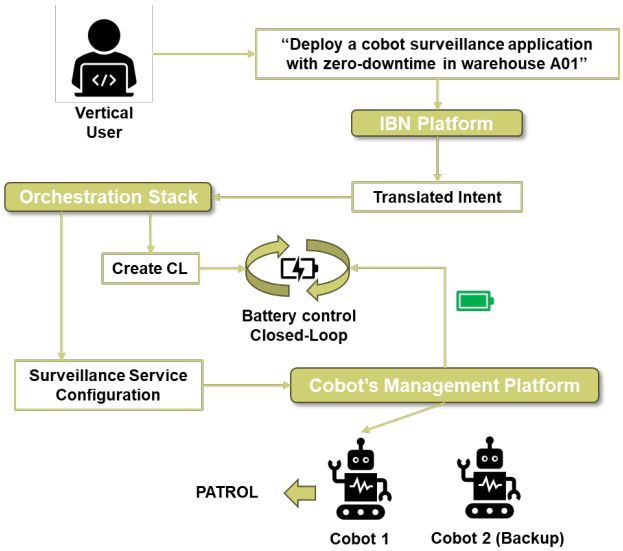

Figure 3-31: Workflow of the intent-based request to deploy the cobot-based application.

When battery charge drops below a given threshold, the CL enforces the migration of the service towards another cobot (cobot2) and the cobots transition to GO CHARGE (cobot1) and PATROL (cobot2) states.

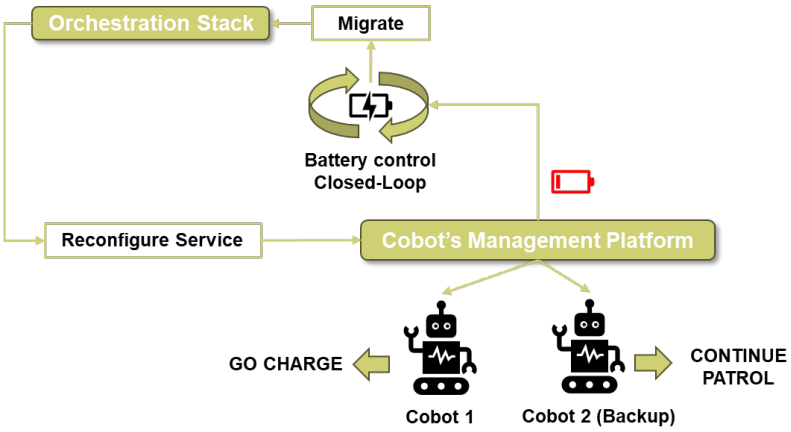

Figure 3-32: Workflow of the Closed-Loop reaction to low-battery level to fulfil the zero-downtime requirement.

3.5 System-PoC #B evaluation results for sustainable and trustworthy 6G systems

This section provides an overview of the performance- and key value- related indicators, results of a selected set of the previous, along with their association to respective service requirements. Given that System-PoC #B deals with evolved versions of System-PoC #A management and orchestration aspects, certain KPIs -related to the aforementioned aspects- remain critical; at the same time, novel KPIs related to the network beyond communications features and intent-based mechanisms must be introduced. The sections that follow include indicators: (i) for the assessment of the performance of the network and application layer components themselves, and various management and orchestration mechanisms; (ii) for assessing aspects related to environmental, social and economic sustainability.

3.5.1 KPIs related to System-PoC #B

The application features of System-PoC #B discussed in the subsection 3.3 reference to various network, compute and application layer KPI requirements. A detailed set of the technical KPIs related to the System-PoCs implementation was listed in D2.2 [HEX223-D22]. Among these, bitrate and E2E latency are particularly crucial metrics due to their significant impact on compute-heavy operations such as inventory-audit and computer vision tasks in real-time.

For instance, the communication between robotic devices, infrastructure sensing devices, and edge / remote backend server application features, can be evaluated through upload and download bitrate as well as network latency measurements. Swarm management operations, which include exposed data processing, path planning and navigation operations, also heavily depend on these metrics. Here, both upload and download bitrate, network latency and edge / remote server processing latency are critical.

Many of the applications feature robotic platform teleoperation, necessitating ultra-low latency between the remote user (tele-operator) and the robotic platforms in the field. Consequently, uplink and downlink network latency become key measurements. Teleoperation also involves computer vision-related operations such as compressing real-time high-definition video streams. This process is influenced by on-board (device) processing latency, the upload bitrate and network latency for transmitting compressed video, and processing latency at the edge / remote server for inferencing and decision-making operations. Additionally, the response (enforcement) back to the robotic devices, affected by download bitrate and network latency, is crucial. Last but not least, service availability (reduced downtimes respectively) is of utmost importance.

3.5.2 Social, environmental, and economic sustainability aspects

System-PoC #B in Hexa-X-II focuses on the feasibility to achieve the target KVIs of social, environmental and economic sustainability. Trustworthiness (as part of the social sustainability key values) refers to the reliability, security and overall integrity of the network and its provided services. Handover of the operations in a general failure, as well as the injection of distributed intelligence and automation characteristics will assure the resilience of the system and thereby enhance the trustworthiness.

Environmental sustainability aspects regard both the operation of the application per se, as well as the operation of the networking and computing infrastructure in a 6G ecosystem. The operations of the cobots with an energy efficient perspective will maximize the lifetime of the operations which will contribute to the environmental sustainability of the system. The improved human-machine interaction in System-PoC #B with the intelligent cooperation among cobots via management and orchestration of the 6G continuum, and resource-usage efficiency will increase the overall environmental sustainability. This will impact the overall operations in the manufacturing process by maximizing the lifetime of the operations in a resource limited environment. In parallel, enforcement of energy-efficient orchestration policies (i.e., dynamic re-allocation of functionality and up/down-scaling of compute resources) in the deployment and lifecycle management of the application can reduce the overall energy consumption across the infrastructure and help to achieve environmental sustainability targets.

The economic sustainability aspects considered within System-PoC #B, include the resilience and reduced downtime, ensuring continuous and reliable service, results have been reported in [HEX223-D22]. Additionally, the energy efficiency aspect of System-PoC #B lowers the operational costs associated with power consumption, whereas the flexible topologies feature can reduce the capital expenditure required for infrastructure development which lower both operating expenses (OPEX) and capital expenses (CAPEX). Together, these elements create a robust economic foundation, supporting the long-term viability and competitiveness of the 6G ecosystem.

3.5.3 KPI- and KVI-driven evaluation results

In this subsection, System-PoC #B results, on the implementation scenarios and PoC's components, discussed in the previous sections, are presented. These results are related to power consumption, E2E application latency, network utilization and service availability, among others.

3.5.3.1 *Autonomous Operation Scenario: Advanced M&O, Flexible topologies and Network beyond communications enablers in Cobot-powered Warehouse Inventory Management –*

The implementation of the scenario on advanced M&O, flexible topologies and network beyond communications enablers in cobot-powered warehouse inventory management, involved the monitoring of various metrics.

Figure 3-33 shows the preliminary results obtained related to trustworthiness (left graph) and energy consumption (right graph) with the utilisation of the functionality allocation algorithm based on the genetic algorithm paradigm, compared to the feasible round-robin placement algorithm (baseline). In particular, the graph on the left shows the percentage increase of trustworthiness obtained by the comparison of the baseline and the functionality allocation algorithm and the graph on the right shows the percentage decrease of the energy consumption accordingly. The functionality allocation optimisation mechanism estimates the close to optimal placement of the computational workloads to the available compute nodes by minimising the weighted (a_1, a_2, a_3) objective function, $\min_{x,z}(a_1 E + a_2 L - a_3 T)$, which consists of the energy consumption term E, the E2E latency term L, and the trustworthiness term T. The trustworthiness term is the sum of the trust indexes of the compute nodes utilised for the workload placement. The trust index of the compute node j is the weighted $(w_1, w_2, w_3, w_4, w_5)$ sum of the availability A_j, reliability R_j, security S_j, multi-connectivity capabilities M_j and battery level B_j, $trustN_j = w_1 A_j + w_2 R_j + w_3 S_j + w_4 M_j + w_5 B_j$. The energy consumption term consists of the processing, transmission and the RF harvester generated energy.

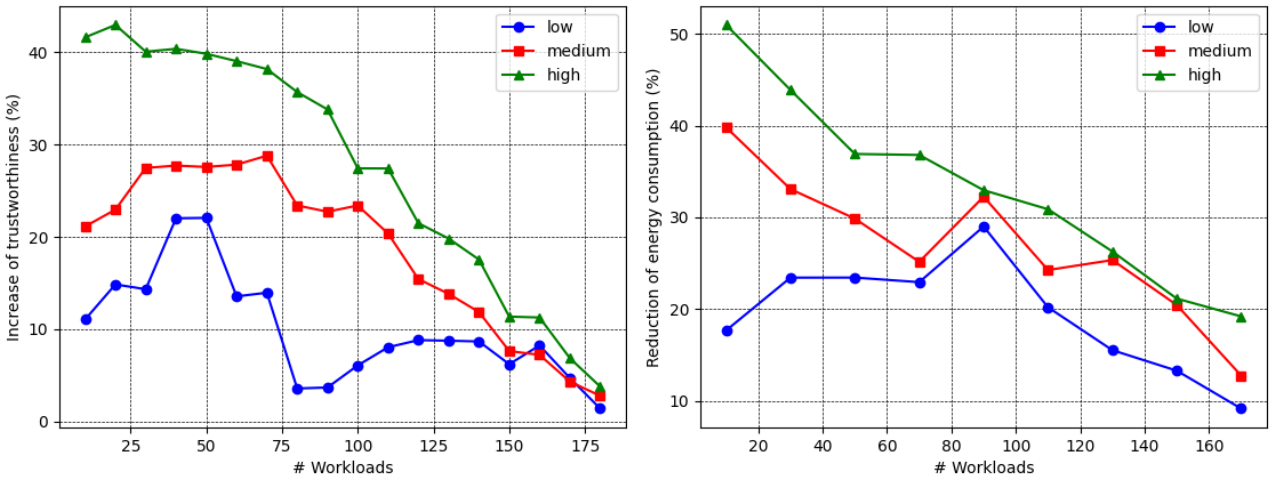

Figure 3-33: Trustworthiness measurements (left graph) and energy consumption measurements (right graph) of the functionality allocation mechanism.

The trustworthiness and energy consumption measurements were taken for different trust weight levels a_3, low, medium and high and energy weight levels a_1, accordingly. These levels correspond to an almost zero contribution, a moderate contribution and full contribution, respectively, compared to the other terms of the objective function. Fifty fixed virtualised compute nodes were used, with various capabilities and an increasing number of compute workloads with varied requirements. A detailed report on the range and values of the parameters used for these graphs can be found in D6.3 [HEX224-D63].

As shown in the graph on the left, the higher the weight a_3 of the trust term of the functionality allocation mechanism used, the higher the percentage of increase of the trustworthiness was obtained which was as expected. Also, the more workloads need to be placed, the percentage increase of trustworthiness decreases. This is because when many workloads need to be placed, the number of feasible placement solutions decreases and the potential gains in the percentage increase of trustworthiness become limited. In general, in this scenario, the metaheuristic functionality allocation algorithm, compared to the baseline (round-robin placement), can gain up to 43% increase of trustworthiness. Moreover, as the graph on the right indicates, the higher the weight of the energy term a_1 is, the higher the gains of energy consumption are, which is as expected. Also, as the number of computational workloads increases, the energy consumption gains decreases. This is because, as was already explained, when there is sufficient availability of compute resources, thus solution space and optimization potential, the described solution provides higher gains. The scarcer those

resources become; the lower gain potential is observed. The energy consumption gains obtained in the described scenario by the functionality allocation algorithm compared to the round-robin placement are 9.9-50.9%.

Next, regarding the trustworthy flexible networks' deployment, the evaluation of UAV power consumption under different scenarios (Table 3-1), as depicted in Figure 3-34, underscores the strategic importance of task distribution between UAVs and edge computing resources in 6G networks.

The graph in Figure 3-34 shows battery depletion for UAVs under three different configurations: moderate transmission load with low central processing unit (CPU) utilization, high transmission load with low CPU utilization, and low transmission load with high CPU utilization. This analysis highlights that processing workloads that require high volumes of data on-board, rather than transmitting data to the nearby edge resources, may extend UAV operational periods. This flexibility is crucial for sustainable 6G-powered operations, allowing UAVs to perform tasks for longer durations without frequent recharges. Moreover, the advanced capabilities of 6G technology enable efficient UAV-to-edge communication, maintaining performance without significantly draining the UAV battery during moderate computational tasks. This balance ensures that UAVs can operate effectively while minimizing energy consumption. By strategically allocating tasks based on UAV capabilities, the function allocation mechanism can optimize resource use, reducing power consumption on compute nodes closer to the cloud and maintaining high efficiency and performance.

Table 3-1: UAV battery test scenarios characteristics.

Conf. #	Setup
1	Transmission Load: 2Mbps, CPU Utilization: 22%, CPU cores: 7, Battery Autonomy: 0.32h
2	Transmission Load: 100Mbps, CPU Utilization: 22%, CPU cores: 7, Battery Autonomy: 028h
3	Transmission Load: 0Mbps, CPU Utilization: 60%, CPU cores: 7, Battery Autonomy: 0.30h

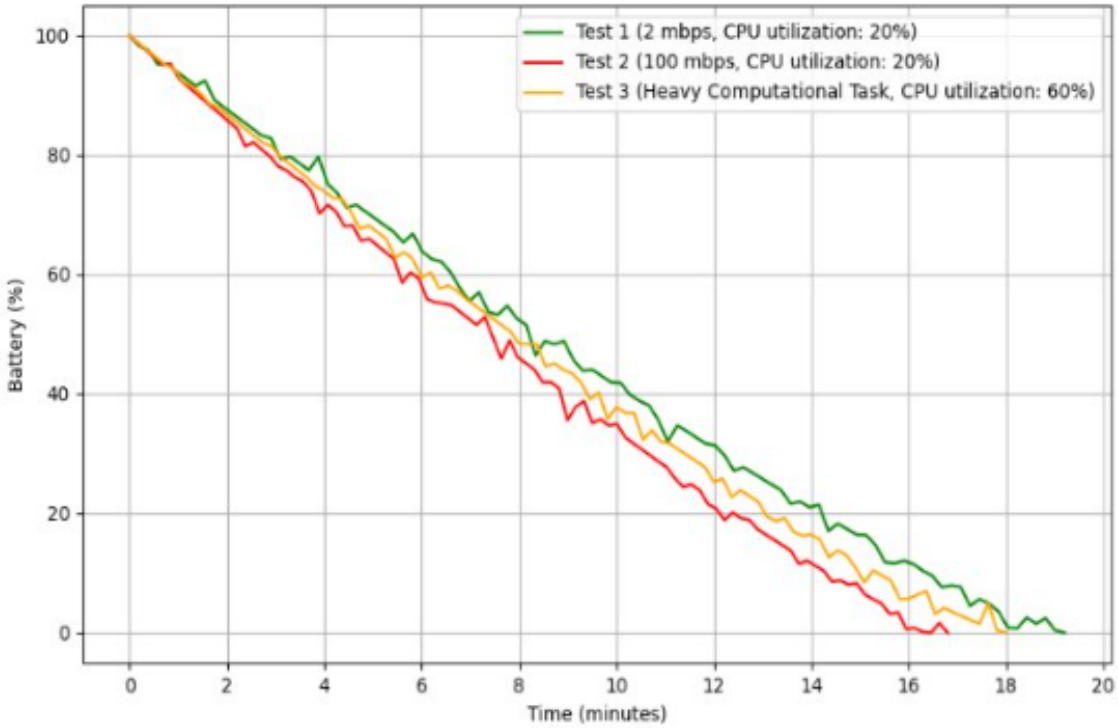

Figure 3-34: UAV battery depletion under different scenarios.

Another set of experiments focused on the power consumption aspects when the AI/ML operations are allocated a) at the edge (compute offloading) and b) at the robotic platform (on-board processing). The following chart in Figure 3-35 provides a comparison of battery usage over time for three different processing

scenarios: on-board processing combined with data transmission, on-device processing only, and offloading processing to the edge (no on-board processing). The red line represents the scenario where both data processing and transmission occur on the device, showing a steep decline in battery percentage, which indicates high power consumption. The blue line, representing only on-device processing, shows a less steep decline, highlighting moderately reduced power consumption. In contrast, the yellow line, which represents offloading tasks to edge computing via 5G communication, shows the slowest decline in battery percentage, demonstrating the lowest power consumption among the three scenarios. Inferring from the chart, edge vs on-board compute in this experiment demonstrated a reduced battery depletion of approximately 20% under heavy ML operations.

For mobile robotics applications that rely on battery efficiency, this approach has substantial impact. Firstly, the significantly reduced power consumption allows devices to operate for extended periods of time without needing to recharge, increasing their cost-effectiveness and operational efficiency. Furthermore, offloading tasks to the edge can enhance device performance by preventing overburdening of the device's processors, leading to faster task execution and better overall performance. Lastly, this approach promotes efficient resource utilization by leveraging edge computing resources, enabling devices to handle more demanding applications and workloads.

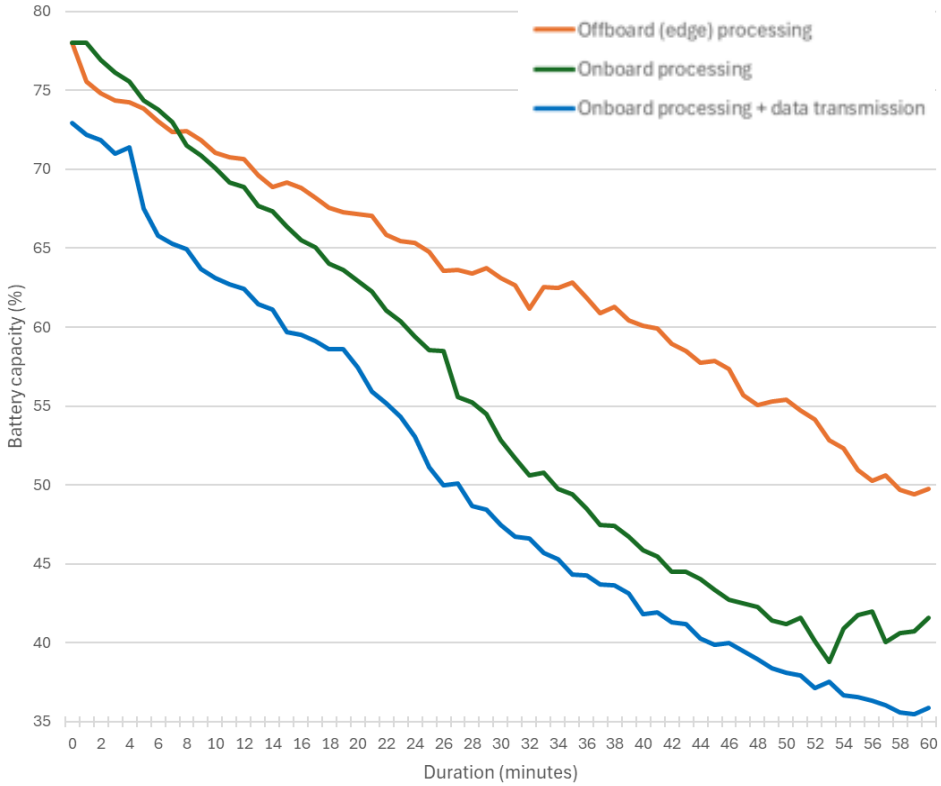

Figure 3-35: Power consumption comparison on-device vs edge computing setup.

Lastly, the effect of offloading ML operations is examined, with regards to CPU utilization and inference time of the ML model. For clarification, the model was deployed with CPU resources on both cases (i.e., on-device processing and offloading scenarios) for the sake of comparison (robots were not GPU-enabled).

In Figure 3-36, the mobile robotic platform's CPU utilization in three different scenarios is illustrated: on-device inference combined with data transmission, on-device inference only, and offloading inference to the edge. The red line represents the scenario where both model inference and video streaming occur on the device, showing the highest CPU utilization and maximizing almost all the robot's processing cores. The blue line, representing only on-device processing, shows a slightly lower baseline with less variability and finally the yellow line, which represents offloading inference to edge computing via 5G communication, shows a significant reduction in CPU utilization as the robot is only publishing the camera data through ROS2.

To get a more detailed insight on the performance of the ML model on the robot vs on edge compute, Figure 3-37 is provided that visualizes the inference time of the model during the time of the experiment. In this chart

we can clearly see much faster and more stable inference performance on edge compute (yellow line), while on the robot (red line) we have an approximately three times increase in inference time and a much more varying performance.

Overall, offloading heavy computational tasks to edge/cloud nodes via B5G/6G significantly improves robot performance and efficiency. Figure 3-36shows a marked reduction in CPU usage from 80% to 30% when AI models are offloaded, while the right chart highlights a substantial decrease in inference time variability, with stable times around 100ms compared to 400-800ms onboard. These benefits demonstrate that leveraging B5G/6G for cloud/edge computing enhances CPU utilization efficiency, ensures faster and consistent processing, and optimizes overall system performance.

Figure 3-36: Performance benefits of offloading computation via 5G on: CPU utilisation.

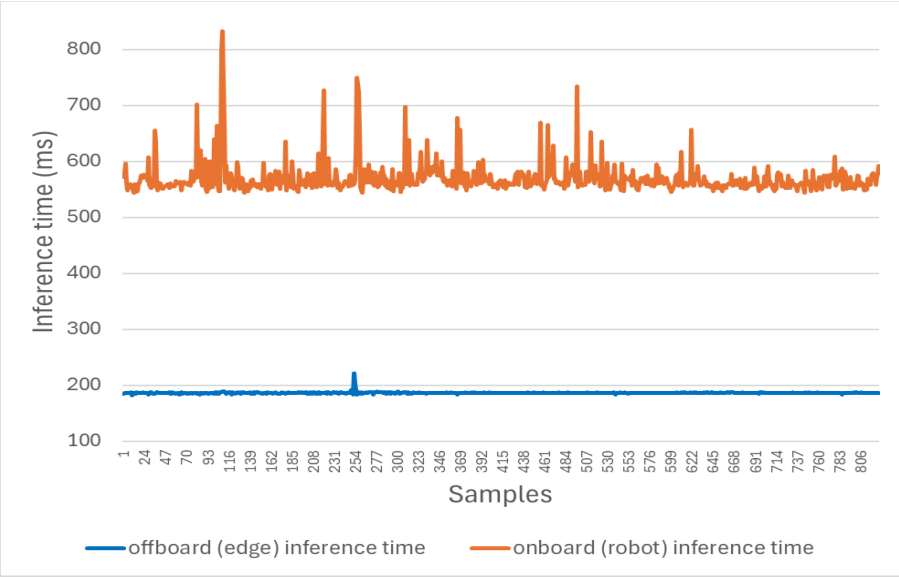

Figure 3-37: Performance benefits of offloading computation via 5G on inference time analysis.

3.5.3.2 Zero-touch Cobot-based video surveillance Scenario: Intent-Based Service Provisioning

As part of the System-PoC #B outcomes in this phase of the Hexa-X-II project, a first set of results from the intent point of view were obtained with the objective to evaluate the latency to deploy the surveillance cobot

application provisioning (3.4.2). To do so, a set of 15 tests were requested using the human based requests such as “create a cobot service on warehouse 24 with zerodowntime”, and as a main result, the corresponding provisioning times were obtained and presented in Table A-1 in the Annex section. Prior to the evaluation of the obtained provisioning time samples, the generated intent and intent report data objects are presented to demonstrate the correct interpretation of the initial human-based request.

In this section the outputs of the system after performing the three main operations for the current IBN-IME solution are presented in order to verify that they follow the 3GPP data model selected [28.312]:

- To deploy and intent, an example of its process logs can be found in Figure A-1.
- To retrieve an intent data object, an example of its process logs can be found in Figure A-2.
- To retrieve an intent report data object, an example of its process logs can be found in Figure A.

After the intent deployment procedure is done, the response used to notify the basic information to the requester is presented in the form of a JSON with the identifiers of the created intent object and its associated report. In there, the requester may identify the name and its status, together with the identifier values to either select the complete intent or the associated intent report.

Using the “intent_id” from the previous JSON, the requester may retrieve the complete intent information. The generated intent data object is illustrated as a JSON data object (below) which follows the 3GPP data model previously presented:


```
{
 "request": "create a cobot service on warehouse 24 with zerodowntime",
 "observation_period": "0",
 "intent_admin_state": "ACTIVATED",
 "user_label": "cobot",
 "intent_id": "ebb3c42e-ef5b-4588-a550-7bbe3c4412f7",
 "name": "cobot_warehouse_24",
 "intent_priority": "50",
 "intent_report_ref": " c10d602f-27a3-41a1-80ee-32ee7dd9fff3",
 "intent_expectations": [
  {
   "expectation_object": {
    "expectation_object_type": "Network Service",
    "expectation_object_instance": "9a294312-ed73-4765-b473-178494c08c34",
    "object_contexts": [
     {
      "context_value_range": "['24']",
      "context_condition": "IS_EQUAL_TO",
      "context_attribute": "warehouse"
     },
     {
      "context_attribute": "soccer_service",
      "context_condition": "IS_EQUAL_TO",
      "context_value_range": "a17b28e4-4fb5-42df-b285-88b66e0c7a37"
     }
    ]
   },
   "expectation_id": "bcdad622-73f8-4787-af09-a0c31ddf203b",
   "expectation_verb": "DELIVER",
   "expectation_targets": {
    "target_contexts": []
   },
   "expectation_contexts": [
    {
     "context_value_range": "563fe86e-00ae-11ef-92c8-0242ac120002",
     "context_attribute": "sla",
```

```
            "context_condition": "IS_EQUAL_TO"
           }
          ]
         }
        ],
        "intent_contexts": []
       }
```


In addition, using the “intent_report_ref” from the first obtained JSON, the requester may retrieve the complete intent report data object, which is illustrated as a JSON data object (below) to demonstrate the use of the 3GPP data model previously presented:


```
       {
        "intent_report_id": " c10d602f-27a3-41a1-80ee-32ee7dd9fff3",
        "intent_ref": " ebb3c42e-ef5b-4588-a550-7bbe3c4412f7",
        "last_updated": "2024-05-30 10:20:13.599878",
        "intent_fulfilment_report": [
         {
          "intent_fulfilment_info": {
           "not_fulfilled_state": "",
           "fulfilment_status": "FULFILLED",
           "not_fulfilled_reason": ""
          },
          "expectation_fulfilment_results": [
           {
            "expectation_id": "bcdad622-73f8-4787-af09-a0c31ddf203b",
            "expectation_fulfilment_info": {
             "not_fulfilled_state": "",
             "not_fulfilled_reason": "",
             "fulfilment_status": "FULFILLED"
            },
            "target_fulfilment_results": []
           }
          ]
         }
        ],
        "intent_conflict_reports": [
         {}
        ],
        "intent_feasibilitycheck_report": {
         "infeasibility_reasons": "",
         "feasibility_check_result": "FEASIBLE"
        }
       }
```


Table A-1 in the Annex section presents the obtained time values during the 15 executed tests related to the three main process to achieve the deployment of the intent: a) vertical service instance creation, b) the NLP processing, and c) intent feasibility. The result in this table shows quite an uneven distribution in terms of time. The biggest part of the request is spent creating the vertical instance, whereas the implementation of the NLP techniques and the feasibility check spend only a few tenth and hundredths of milliseconds respectively.

To complement the information, a Cumulative Distribution Function (CDF) graph for the time spent on the three variables is showed. The CDF graphs (Figure 3-38, Figure 3-39, Figure 3-40) represent the probability that a given process requires less time than (or equal time to) a certain value, illustrating how the data points accumulate over the range of values:

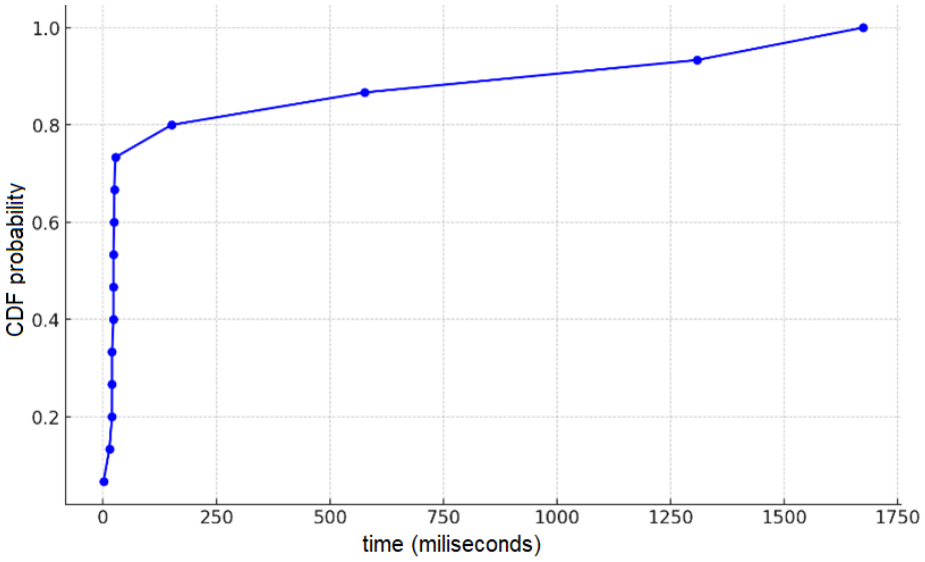

Figure 3-38: Vertical service instance creation process CDF.

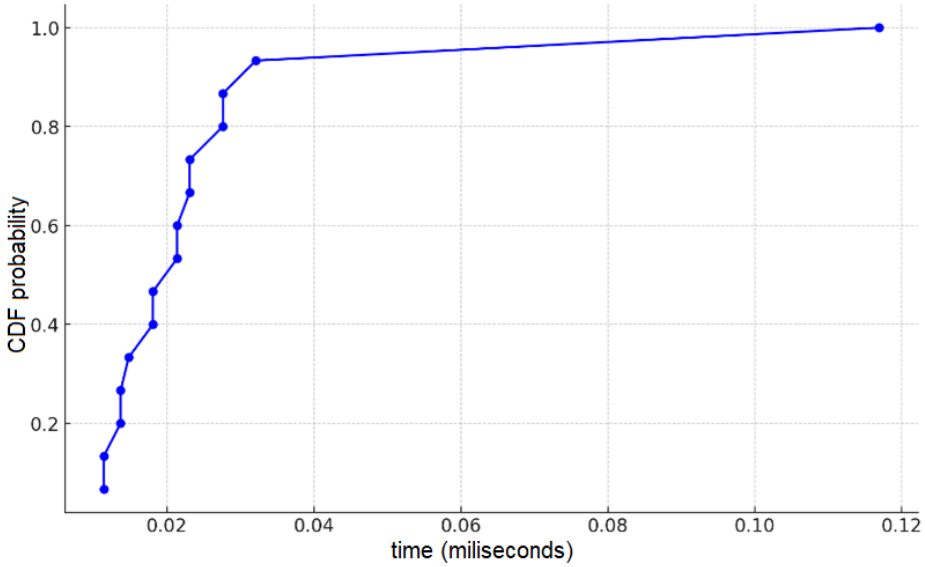

Figure 3-39: Natural Language Processing (NLP) process CDF.

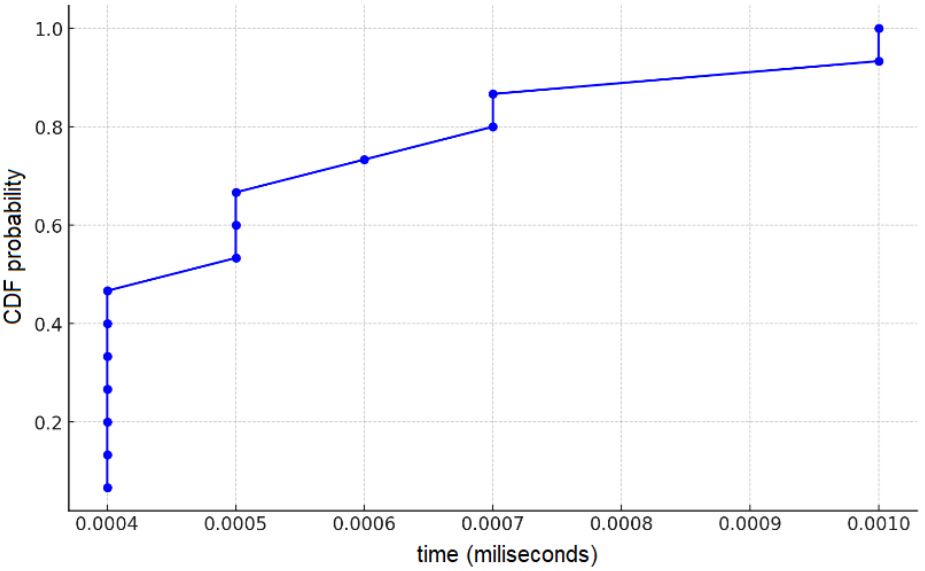

Figure 3-40: Intent feasibility process CDF.

As it is possible to see, when creating a vertical instance, which is the action that takes most of the time, most of the instances present a uniform distribution with the presence of a few outliers that took meaningfully more than expected. A similar picture is also shown with the CDF of the time spent applying natural language. However, it is worth highlighting that the scale of this graph on the X axis is much lower than the previous one which implies a meaningfully lower amount of time taken to complete the task. The feasibility check, however, is the variable that presents a more uniform distribution of its values.

The presence of outliers when calculating the time spent for the vertical instance can be easily seen in a histogram created on Figure 3-41, where it is possible to see that 11 instances out of 15 are within a range of 0 to 250 seconds, leaving only 4 instances with a meaningful bigger amount of time.

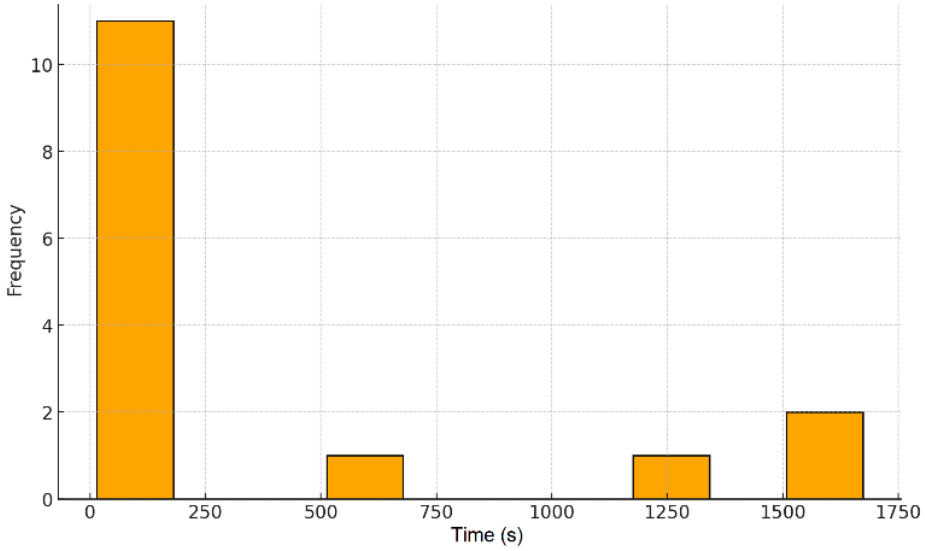

Figure 3-41: Intent deployment time distribution histogram.

3.5.3.3 Zero-touch Cobot-based video surveillance Scenario: Integrated Closed-Loop for Cobot Service Migration

During the trials on Intent-Based Service Provisioning with Integrated Closed-Loop for Cobot Service Migration, network latency and throughput metrics were monitored. These metrics provide insights to whether the KPIs identified for this use case are fulfilled. Among the KPIs identified are the reduction of the service zero-downtime and resource usage optimisation in the cobot surveillance service. Additionally, the network QoS is evaluated since it's a key component provisioning these services.

A simplified figure (Figure 3-42) of the measurement setup for measuring latency and throughput metrics is depicted below:

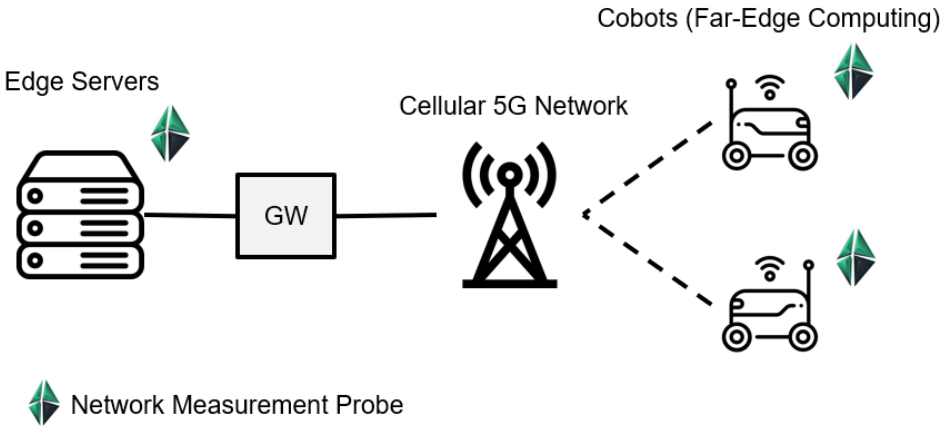

Figure 3-42: Measurement setup for the cobot's application latency and throughput metrics.

In this setup there are 3 measurement endpoints. On one side, the cobots, equipped with webcams, generating a Real Time Messaging Protocol (RTMP) video surveillance stream, and the Edge Servers which host cobot services such as the RTMP servers for the intent consumers to view. In between these endpoints there is a 5G Stand Alone (SA) Network composed of Nokia ASiR RAN elements and a Open5GS core and a Gateway (GW) towards the Edge Servers.

Network QoS measurements were performed from these endpoints beforehand to evaluate the networks throughput and latency capabilities. These measurements consisted of using tools, ping and iperf3, to generate traffic. Results were obtained with Qosium measurement tool [Kai23]. The outcomes are depicted in the figures below (Figure 3-43, Figure 3-44 and Figure 3-45).

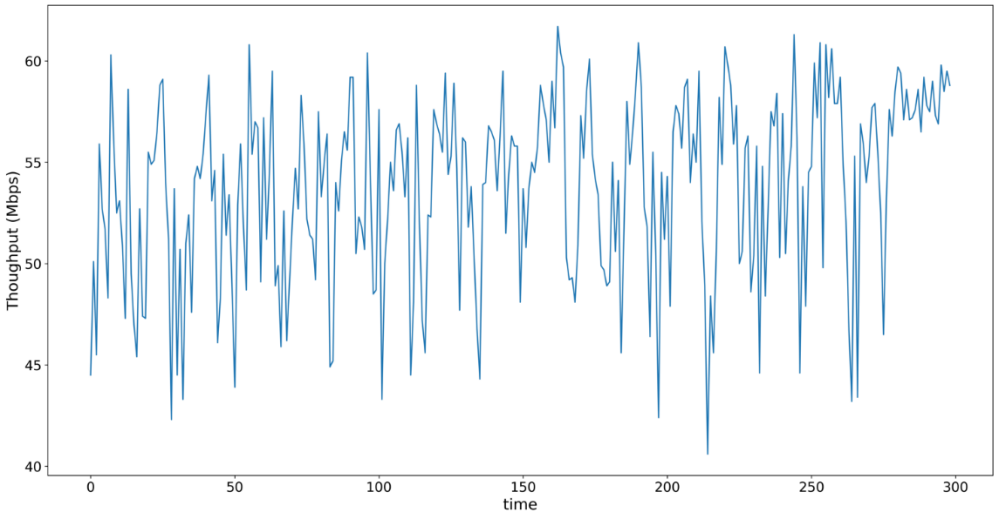

Figure 3-43: The Cobots Uplink Throughput in a 5G SA network.

The TCP iperf3 uplink measurement, Figure 3-43, obtained 300 samples, and yielded the following results. The median value of the measurement is 54.70 Mbps. The figure shows noticeable fluctuations in the throughput measurement. Nevertheless, the results show that the platform is capable of transmitting video streams with sufficient frame rate and resolution for the surveillance mission use case.

The ping measurements, Figure 3-44, obtained 300 samples, and yielded the following results. The median of the measurement is 10.1 ms. Fluctuations are present but not significant, the mean value is 10.49 ms and the maximum is 17 ms. Overall, the round-trip time values are stable, and the system can host low latency applications.

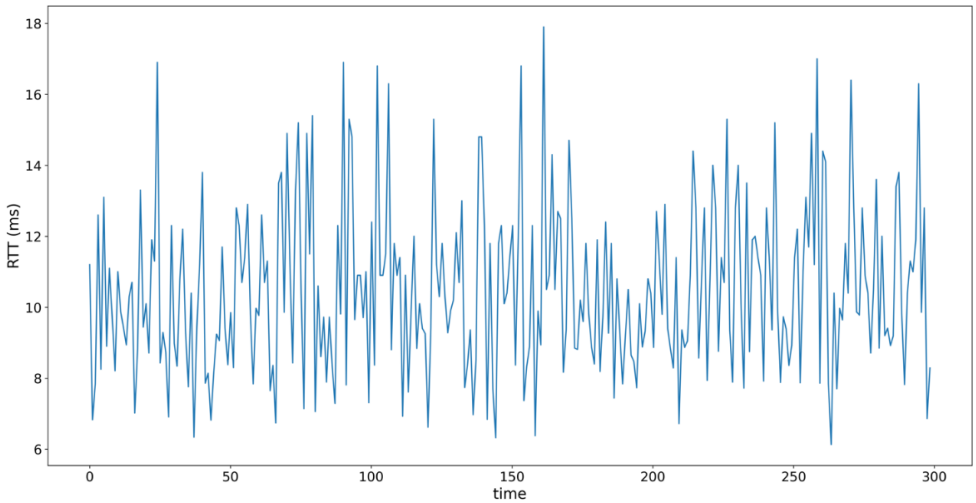

Figure 3-44: Cobots Round-Trip Time (ms) to the Edge Servers, in a 5G SA Network.

In the application migration test, service downtime was monitored by collecting metrics from the traffic generated from the video streaming application. Where the traffic generated from robot 1 is shown in blue and

the one coming from robot 2 is in green. As Figure 3-45 illustrates their transmission times overlap by 30 seconds.

For a better comprehension of the figure outcomes, the vertical lines indicate the time actions that are triggered by the CL and when the video stream is visible to the intent consumer. Moreover, the time series data represents the transmission of video datagrams from robot 1 (blue) and robot 2 (green) to the RTMP video server.

When robot 1 transmits the video stream at first the initial video stream starts about 10 seconds after the service pod is deployed on the cobot node. Afterwards when the replacement is triggered the pod takes 30 seconds to terminate while the video service in the replacement cobot has already started. The pod termination is triggered by the kubernetes API, one possible explanation for this behaviour is that the application tries to terminate the video stream gracefully by waiting to empty the video buffer. Additionally, the video stream from the second cobot takes 35 seconds to start since the transmission starts. This degrades user experience, and we consider studying this behaviour and improving it in future work by studying the video application stack in detail.

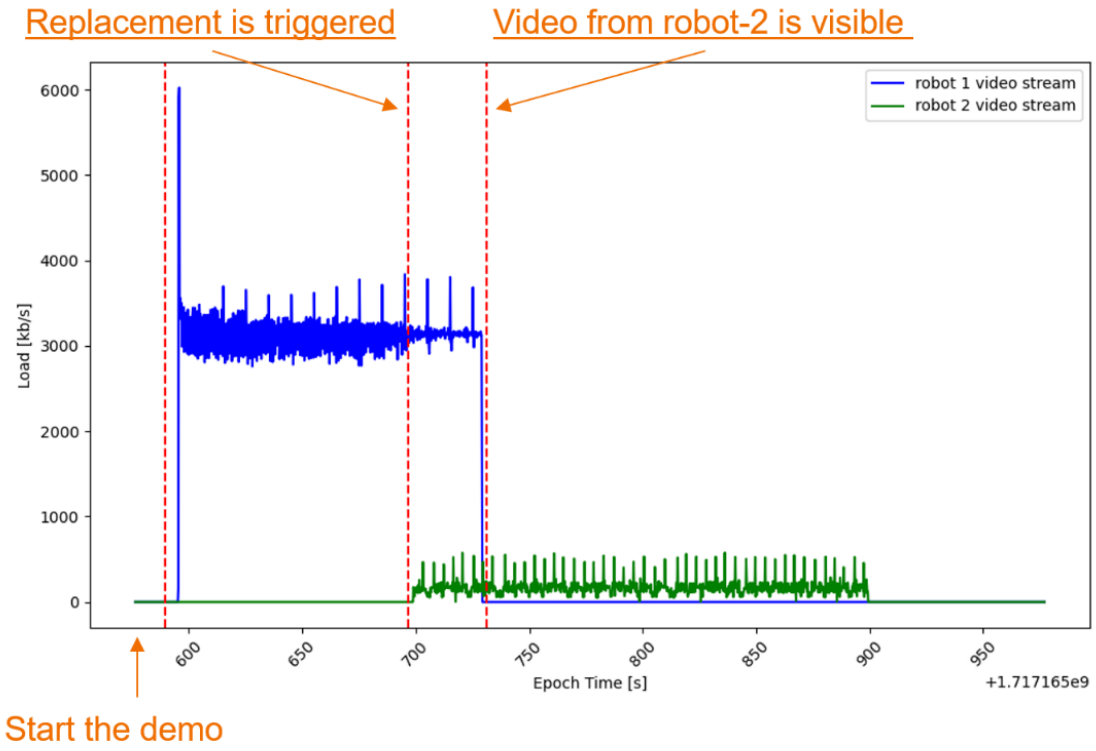

Figure 3-45: Performance evaluation of a surveillance service migration.

3.5.3.4 *PoC Component: AI-assisted E2E lifecycle management of a 6G latency-sensitive service across the compute continuum*

As described in Section 3.3.1, the incoming workload is divided in low/high values depending on the state the application is at. In specific, low workload values are observed during normal operation, while increased values are measured when there is a warning. A pulse demonstrating low and high values of the workload is shown in Figure 3-46.

In this example, we identify the different behaviours that the application showcases across the Edge/Cloud infrastructure used. The deployment of the application and the experiments are conducted on a multi-cluster testbed over a virtualized infrastructure, composed of two clusters on two *OpenStack* Virtual Machines (VMs) in two Intel Xeon Silver 4110 servers. The first is a high-resource Kubernetes cluster (referred to as the "cloud cluster") with a capacity of 16 vCPUs and 16 GiB of RAM. The second is a resource-constrained, single-node *k3s* cluster (referred to as the "edge cluster"') with a capacity of 4 vCPUs and 8 GiB of RAM. These two clusters are coordinated by the *karmada* management system, which enables multi-cluster/cloud deployments, scheduling, and connectivity. An observability server has been deployed to provide real-time

access to the collected metrics, which are utilized by the intelligent components of the orchestration mechanism for their operations.

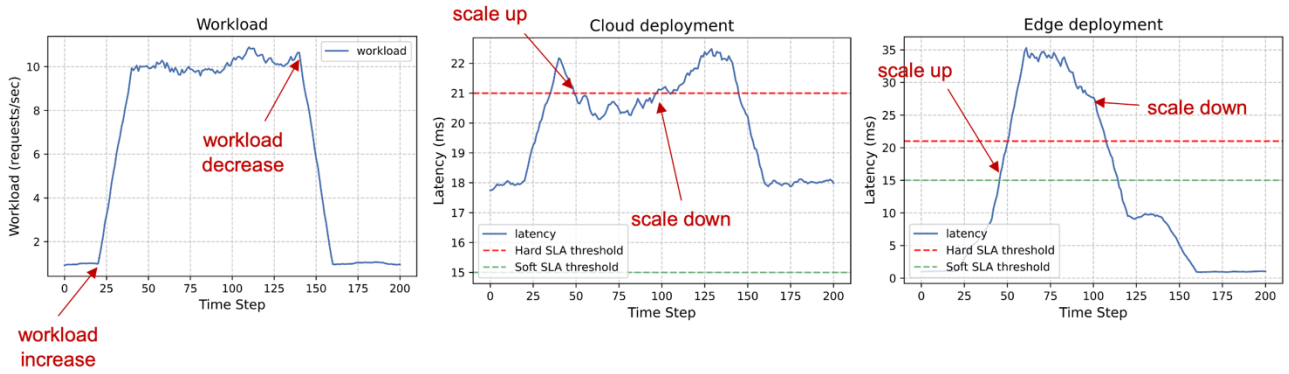

Figure 3-46: Workload (requests/sec) and latency monitoring example.

We experiment with the two different clusters and how horizontal scaling influences performance. As shown in Figure 3-46, none of the machines can achieve optimal performance for all kinds of workloads. The following situations are identified:

- Cloud deployment: When the service is executed at the cloud, scaling up successfully reduces latency by 10%. However, requests sent to the cloud suffer from additional communication latency delays (~15ms).
- Edge deployment: At the Edge, communication latency is lower, but scaling up actually increases latency (up to 3100%), since not enough CPU is available, and concurrency hinders performance severely.

Thus, a hybrid deployment can tackle such a trade-off, working at the Edge during normal operation times, taking advantage of the directly available smaller resources and offloading high workloads to the cloud where scaling up is efficient. For this purpose, in this PoC scenario, a joint autoscaling and offloading mechanism was used. The developed RL-based agent was trained (Figure 3-47) at the aforementioned testbed for learning the problem's specific parameters and identify optimal scaling-placement combinations. In Figure 3-47, the agent's reward during training is illustrated for 2000 steps (ranging from -100 to 100) and is shown to increase up to a certain point where a maximum return (~67/100) is reached, and the algorithm converges.

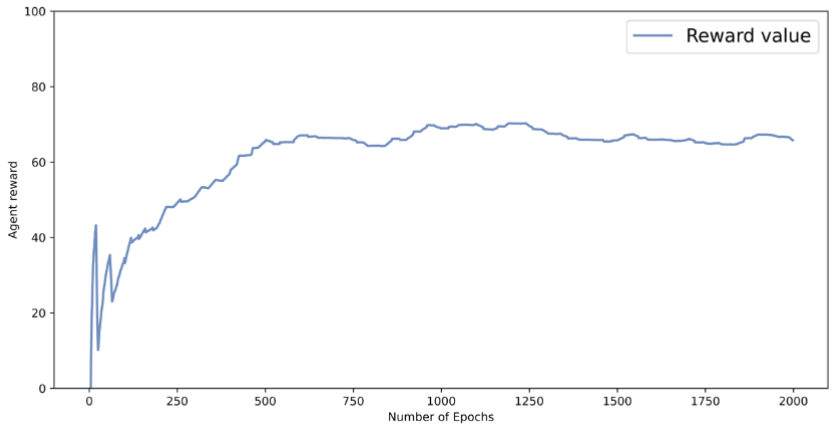

Figure 3-47: RL autoscaling agent training: reward evolution across the first 2000 episodes.

The main aspects being tested and validated are the multi-cluster deployment of the application, the optimized service placement, and the joint orchestration mechanism including RL-based service autoscaling and computation offloading. As described in the PoC scenario, two different workload levels (low/high) are studied based on the application's state machine. For the specific experiments, the state of the agent is constituted by the replicas deployed for the service (1-3 replicas), their placement (edge=0 or cloud=1) and the current workload. The agent is responsible for deciding the replica number suitable for the deployment and whether these replicas should be deployed at the cloud or the edge.

The complexity of the described problem comes from the fact that the identification of the correct number of replicas for the edge server is not straightforward, since such servers have by design low resources to support high numbers of replicas and increased scaling may cause performance deterioration. E2E latency of a service incorporates communication as well as computation latency. Executing a service at the cloud introduces higher communication time, but the average computation time when more replicas are available is decreased. Thus, the agent needs to identify the optimal point to which to increase the replica number of the service without decreasing performance, while considering the difference in communication time in each case. In Figure 3-48, the mechanism's decision-making results are demonstrated.

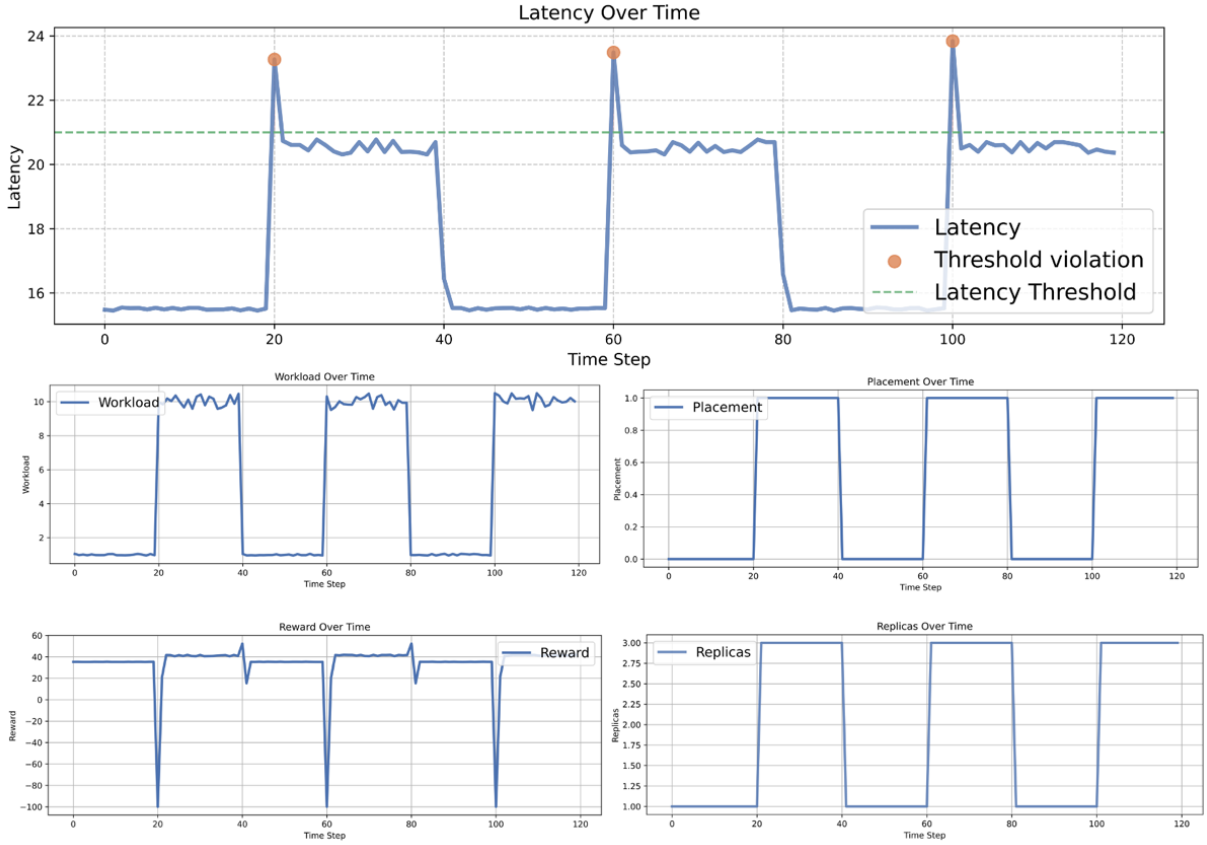

Figure 3-48: RL agent autoscaling performance: workload, decisions, SLO satisfaction, agent reward.

The described workload is given as input in the form of a pulse, which is what is expected based on the specific scenario. The agent learns how to optimally combine the placement and scaling decisions and identifies that for low workloads, placement with 1 replica at the edge (placement=0) provides lower latency, while high workloads demonstrate additional stress and need to be deployed at the cloud (placement=1) with 3 replicas. The deployed version of the algorithm is reactive, meaning that it does not consider future values of the workload and thus, demonstrates a delay in optimal decision making. This is obvious in Figure 3-48, which illustrates that latency is below the SLA for all cases apart from the workload level shift points where unanticipated workload increase causes latency increase, which is regulated in the next steps.

3.5.3.5 *PoC Component: Training and inference of collaborative distributed machine learning model on a dynamically changing heterogeneous 6G architecture environment*

As described in Section 3.3.7, input nodes, generalization node, and the output nodes are deployed as separate Kubernetes nodes for the purpose of demonstrating that they are isolated local entities each having their own local datasets and workloads that are not shared in between. Figure 3-49 is a snapshot presenting the deployed Kubernetes pods for each neural network: one emulating the generalization NN at the core network; three input nodes emulating NNs in the gNB, MAC and UPF respectively; and two output nodes emulating NNs for the video bitrate and delay estimation tasks.

```
vertical-fl-generalization-corenetwork-9rdjg   1/1     Running   0          118s
vertical-fl-input-gnb-tzsqz                    1/1     Running   0          86s
vertical-fl-input-mac-ssdn4                    1/1     Running   0          85s
vertical-fl-input-upf-cgq4k                    1/1     Running   0          85s
vertical-fl-output-bitrate-estimation-krkxz    1/1     Running   0          2m23s
vertical-fl-output-delay-estimation-66klw      1/1     Running   0          2m24s
```

Figure 3-49: PoC consists of 3 Kubernetes pods for input nodes, 1 pod for generalization node, and 2 pods for output nodes.

The results related to the properties such as cross-layer training, generalization and multi-task learning of the component PoC #B.2 was previously presented in Section 8.1 in Deliverable 3.3 – Initial analysis of architectural enablers and framework [HEX224-D33]. Therefore, this report presents the results related to the energy consumption and accuracy observed at the output nodes during model layer offloading. The initial description of model layer offloading in the context of vertical federated learning is given in Section 3.3.7 and in Figure 3-25.

The experiment scenario that was performed with the PoC#B.2 is as follows. In the initialization phase, the generic model had a 4-layer NN model, and the output nodes had 1-layer NN models each. During the training 2 last layers from the generalization node is delivered to the output nodes and prepended to the neural networks at 21.5s. The training continues without any interruption in runtime. In Figure 3-50 (left), the transferred information size from generalization node to the output nodes are depicted. The first spike at 21.5s depicts the size of the offloaded model layers. The accuracy is given in Figure 3-50 (middle) and this offload operation did not impact the accuracy. The reason is that the same weights that were originally at the last 2 layers of the generalization node are now attached as the first two layers of the output nodes, hence there was no change in the total model weights. The estimated energy consumption increased from 5 joules to 22 joules due to the increased computation at the output nodes as illustrated in Figure 3-50 (right). The training continued over iterative rounds until 53.4s, and then the layers that were previously offloaded are offloaded back to the generalization node. At this point, the accuracy is impacted due to the aggregation of model layers from two output nodes to the generalization node. This offloading caused the estimated energy consumption at the output nodes to drop from 22 joules to 5 joules. For the demonstration purposes the layers are again offloaded back after a few more rounds of training at 92s from generalization node to the output nodes. Again, as the same neural network layers are offloaded the accuracy was not impacted. In the later rounds of training after ~100s, the accuracy of both output nodes increases simultaneously demonstrating the multi-task learning via split learning. Generalization NN model learned to generalize its activations to two different output NN models that are responsible for two different tasks: video delay estimation and video bitrate estimation.

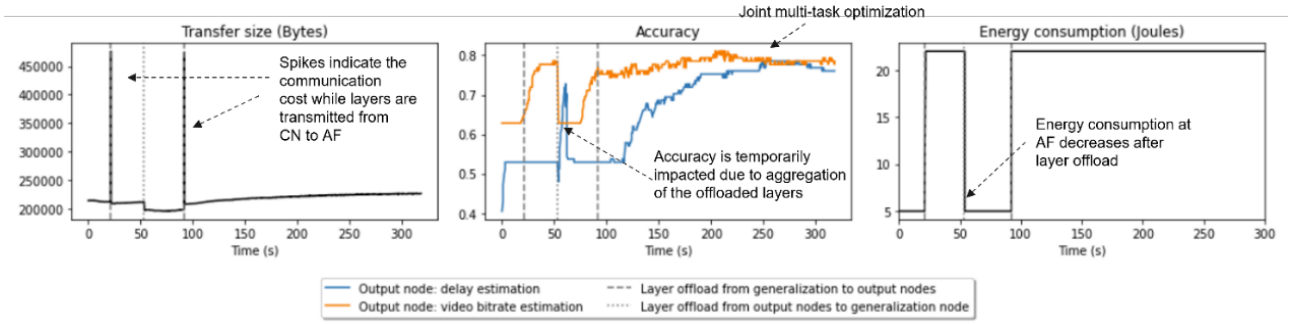

Figure 3-50: Transferred information size from generic node to the output nodes (left), accuracy observed at the output nodes (middle), and the estimated energy consumption at the output nodes (right) are given.

3.5.3.6 PoC Component: Programmable and flexible network configuration

The experimental assessment of the Transport Data Plane-in-a-box with TFS controller provides a clear demonstration of the system's capabilities in handling complex network management tasks. The emulated testbed is illustrated in Figure 3-51. The TFS controller Release 3 was deployed on top of a Virtual Box virtual machine equipped with 4 vCPUs, 12 GB of RAM, and 100 GB of disk running Ubuntu Server 22.04 LTS. To accomplish it, a Kubernetes Container Orchestration system was used to deploy TFS; in particular, the

Canonical MicroK8s v1.24.17 package. As for the TFS deployment specifications, we activated only those relevant components highlighted previously. Together with the TFS controller, the VM runs the packet and optical data plane layers. The packet layer is emulated using ContainerLab with emulated packet routers using freely available gNMI/OpenConfig-enabled Network Operating Systems (SR Linux) and Arista (cEOS). The optical layer is managed through a proprietary Open Line System (OLS) controller emulating the underlying optical network and exposing a ONF Transport API NBI.

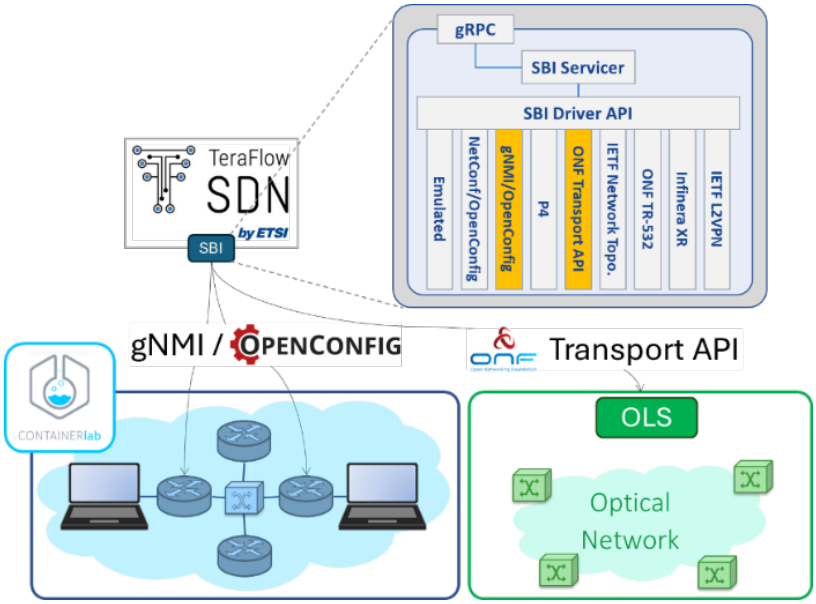

Figure 3-51: Testbed Architecture for DataPlane-in-a-Box.

The assessment is carried out in three main steps designed to thoroughly evaluate TFS' capacity to manage and configure a packet-optical network environment, which can serve as a dynamic programmable backhaul transport network for 6G. First, we carried out the onboarding of the network topology by uploading, through the WebUI, a JSON descriptor file that enumerates the packet routers and the OLS controller together with their management credentials and the driver settings, and the links between them. The resulting topology in TFS is showcased in Figure 3-52, where the optical layer equipment is abstracted by the OLS controller. Following the topology onboarding, a connectivity service was provisioned. This step tested the TFS' ability to handle service provisioning requests by expanding them into underlying services which were then applied to the network elements. The controller automatically configured the network elements using the gNMI/OpenConfig protocols for the packet routers, and the Transport API (TAPI) was utilized to configure the OLS.

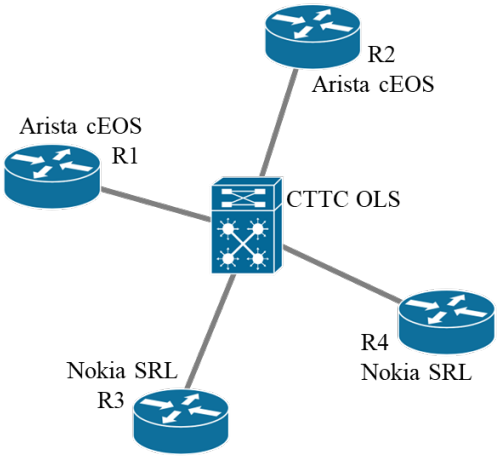

Figure 3-52: Network Topology in TeraFlowSDN.

As a last test, we validated the connectivity through the emulated packet layer devices. Note that the focus of this work is on the control and management plane; emulating electrical/optical/electrical conversions is out of the scope, thus we bypassed the optical layer in this test. For this very reason, we assumed the packet-layer links were supported by optical connections and deployed them as direct links in ContainerLab. A standard network testing tool iperf was used to verify the basic connectivity of the service and to generate some traffic to validate the gNMI-based telemetry stream data collection. Figure 3-53 illustrates a) the connectivity and emulated network throughput measured with iperf, and b) the Grafana plot illustrating the measured traffic capacity. Note that iperf measured ~28 Mbit/s which approximately matches the ~3.5 MByte/s reported in the Grafana dashboard.

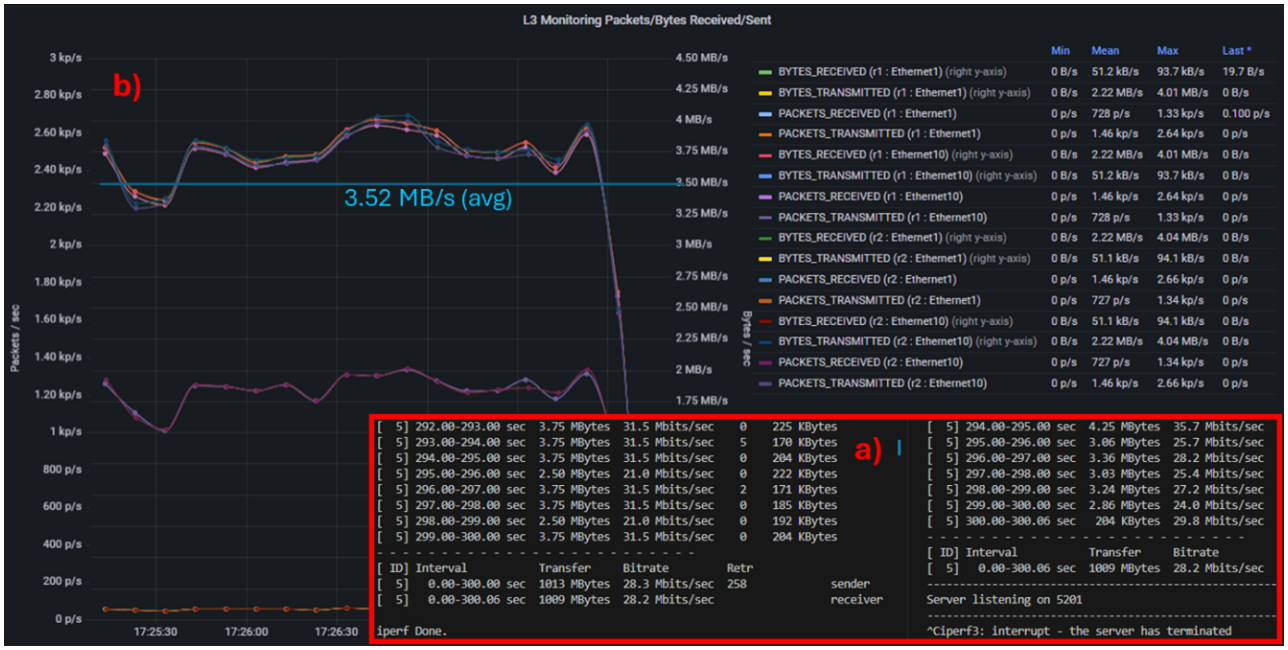

Figure 3-53: Connectivity (a) and Performance Measurement (b) using DataPlane-in-a-box.

3.6 Results on E2E system evaluation by simulations

In addition to the PoC based evaluations, and in order to prepare the path towards the System-PoC #C, some simulation studies were performed to support E2E performance evaluations and complement the current results from the System-PoC #B. Especially, the simulations focus on the 6G RAN and radio enablers which affect to communication performance. The detailed topics of simulations are selected based on the proposed enablers during the project progress. Initial results of latency study done to compare disaggregated and monolithic RAN approaches will be introduced in this subsection.

3.6.1 Simulation Scenario

A simulation study that has been done for latency evaluation of 5G disaggregated RAN and its comparison to monolithic RAN architecture will be introduced here. Namely, in the 3GPP specification of the 5G architecture, the RAN gNB was split into three separate logical nodes (Central Unit (CU) – Control Plane (CP) (CU-CP), CU – User Plane (UP) and Distributed Unit (DU)) [38.401], which is called a higher layer split (HLS), with standardized interfaces in between (e.g., F1 between CU and DU, E1 between CU-CP and CU-UP, see Figure 3-54). HLS connection link (F1 interface) between DU and CU is also called “midhaul” in the RAN architecture. Interfaces were assumed to be simple and latency-insensitive, with the goal to be suitable for multi-vendor deployments and enabling to place the DU and CU to different physical locations. However, the problem of the split between the tightly coupled CU-CP and DU is that neither of the two nodes have all required information to choose an optimal configuration for the UE.

There are different options proposed for the functionality (protocol layers) split between CU and DU. Split Option 2 is recommended by 3GPP for 5G architecture [38.401, 38.801, LCC19]. In the split Option 2 case,

the CU hosts the network layer Radio Resource Control (RRC) functionality and Packet Data Convergence Protocol (PDCP) functionality from the data link layer, while DU includes RLC, MAC and PHY layer functions, as illustrated in the Figure 3-54. Therefore, the control plane split between CU-CP and DU increases the complexity by additional control signaling messages, which causes increased latency and jitter. Moreover, when the CU and DU are deployed in different physical locations, they have separate clocks which needs to be synchronized.

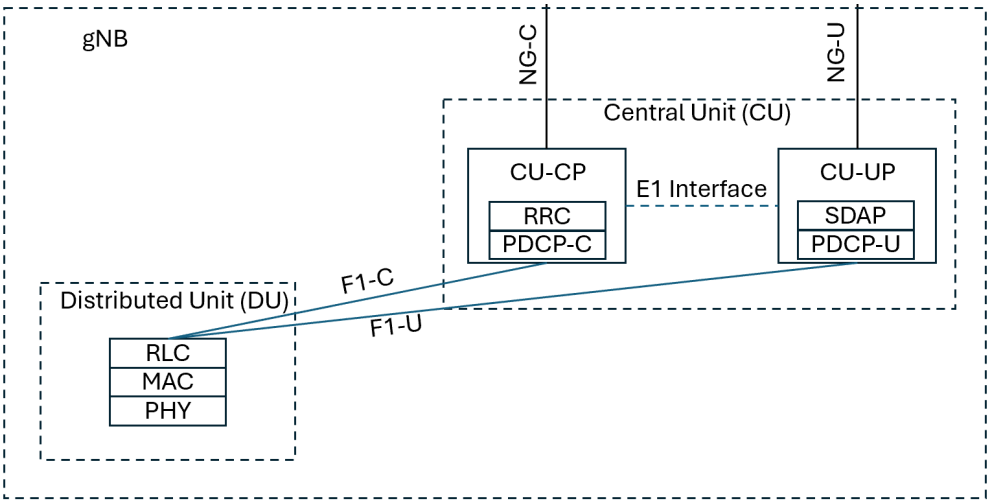

Figure 3-54: Split (3GPP Option 2) of DU and CU in the disaggregated 5G RAN gNB.

If there would not be the HLS between DU and CU, it would most likely decrease the latency and make the optimal network resource configuration for UEs simpler, since each UE would be controlled by a single RAN control function. That would simplify standardization and the implementation of UEs and networks, as well as reduce the number of signaling messages required to acquire and utilize all relevant information for each UE. The lower layer split (LLS), which is called as fronthaul, is seen useful option for 6G since it enables to place Radio Units (RUs) closer to user while the RAN baseband can be centralized in the cloud [HEX23-D53, HEX224-D33]. LLS involves separation of baseband functions and RU by a fronthaul interface, typically based on a functional split between the MAC and physical (PHY) layer, or inside the PHY layer. At the moment, split Option 7.2 has gained most attraction for LLS since it provides a good solution for tradeoff between RU complexity, fronthaul bandwidth and latency requirements and inter-cell cooperation. Open RAN (O-RAN) alliance has adopted a combination of split 7.1 and split 7.2, which is called as a split 7.2x [ORA-24, ORCUS24]. Figure 3-55 illustrates the disaggregated 5G RAN and the considered RAN approach for 6G, as well as HLS and LLS options, which have been proposed up to date [38.801, LCC19, PSG+23, HEX224-D33].

As can be seen from Figure 3-55 there are various split options, which requires comprehensive studies to be able to optimize the trade-offs between complexity, energy consumption, cost, and communication performance [LCC19, PSG+23], as well as to decrease the amount of split options that will be finally standardized for 6G RAN. In [MRT23] has been done latency and jitter evaluation for HLS approach using soft–real-time testbed with functional split between PHY and MAC layers which are run at DU and CU, respectively. It is found that this approach is insufficient for the use cases which have very tight synchronization and latency requirements, e.g., ultra-reliable communication applications. Analytical comparison of monolithic RAN and disaggregated RAN delay has been done in [HEX23-D53]. Authors consider same protocol delay for RAN components, but additional delay is assumed for the disaggregated RAN physical distance separation with fiber connection between the nodes. Theoretical model has been introduced in [DAK+21] for E2E delay evaluation of fronthaul link when using different functional splits. According to our best knowledge, HLS interface message exchange delay evaluation results for a real-time end-to-end (E2E) virtual network has not been reported in previous works.

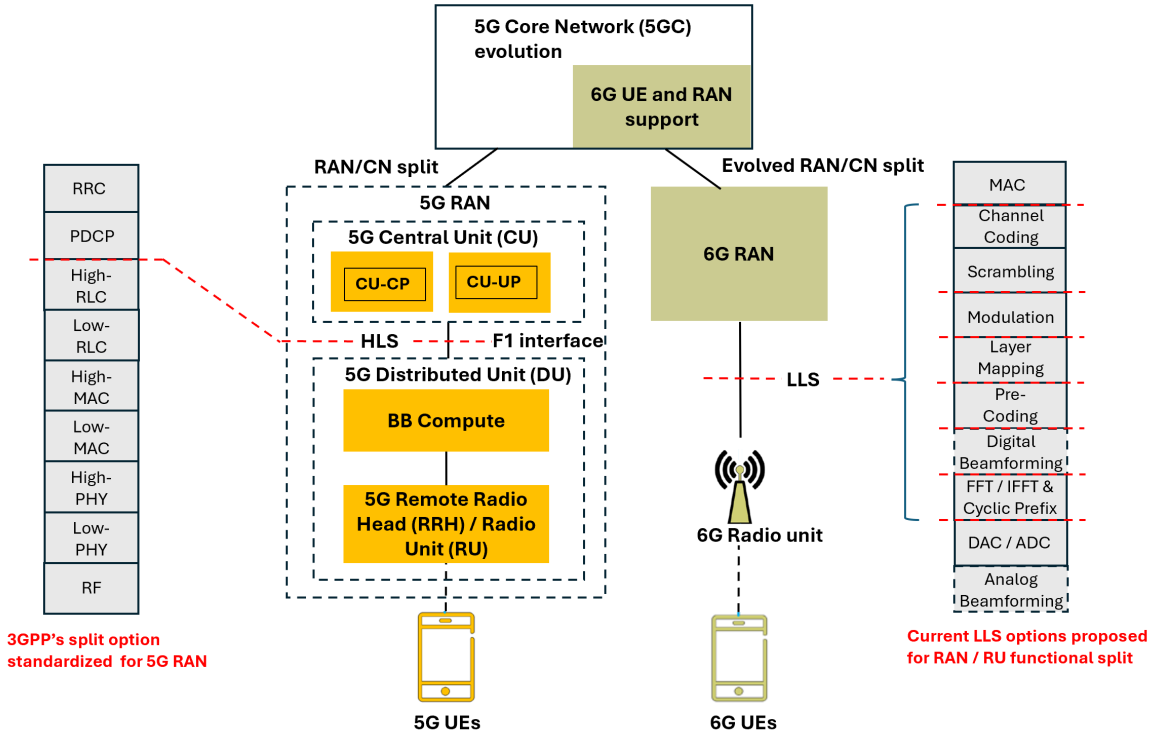

Figure 3-55: 5G and 6G RAN approaches, and HLS / LLS options illustrated.

Therefore, in this work will be performed E2E simulations to study the communication latency of different RAN architecture options, to support selection of suitable split option(s) for 6G system. Simulations are performed by using CU/DU split with F1 interface, and by using monolithic gNB, i.e., without using the F1 interface between CU and DU. RAN architecture design selection will affect directly on the 6G radio enablers performance evaluation, e.g., in the case of distributed-MIMO and handovers between UE and DUs, the RRC protocol has crucial role. Therefore, the study of split options is useful to be implemented to the simulator before advancing to 6G radio enablers performance evaluation.

In this latency study, the E2E simulations will then use two different RAN architecture alternatives, which are illustrated in Figure 3-56 as a part of simulator high-level architecture used in this study. Figure 3-56 illustrates that Alternative 1 is the case without HLS and F1 interface, while Alternative 2 is the 5G approach with the F1 interface between DU and CU. In the simulations, the communication is established between CN and UE, and the latency is studied between RAN and UE for both options. More specifically, the time consumed for F1 interface connection establishment and communication between DU and CU, as well as RRC functions execution time, are studied when the UE initial access is established to the network.

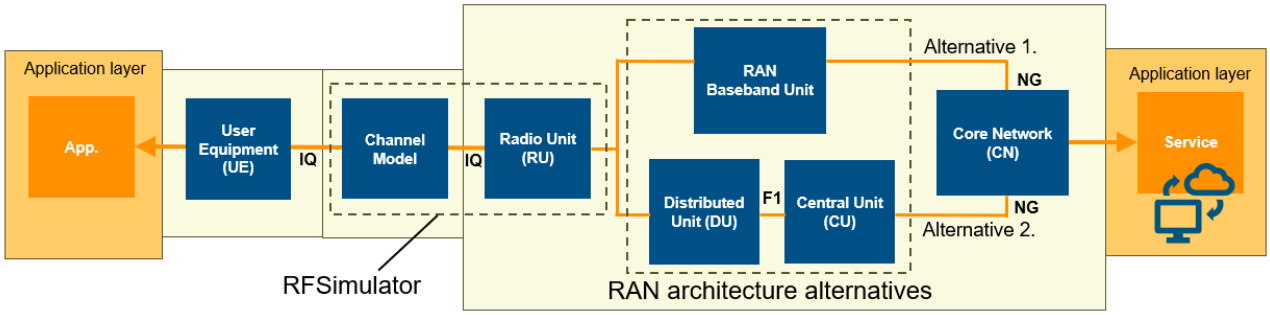

Figure 3-56: High-level architecture of simulator with the studied RAN split alternatives illustrated.

Figure 3-57 shows the message exchange for setup of F1 interface between DU and CU, which is required in the disaggregated RAN case. Figure 3-58 shows the message exchange for the UE initial access procedure for the disaggregated RAN.

As can be seen from Figure 3-57 and Figure 3-58, multiple messages must be exchanged between DU and CU before the RRC connection and UE context setup has been established. In the "Simulation Results" section will be shown simulation results about the time duration of that message exchange procedure, to find out amount of additional latency due to F1 interface between DU and CU. Latency results are compared to the monolithic case, which does not require the message exchange through F1 interface. The E2E simulation platform is based on the open source 5G RAN and core implementations published by Open Air Interface (OAI) community [OAI24, OAI-COR, OAI-RAN].

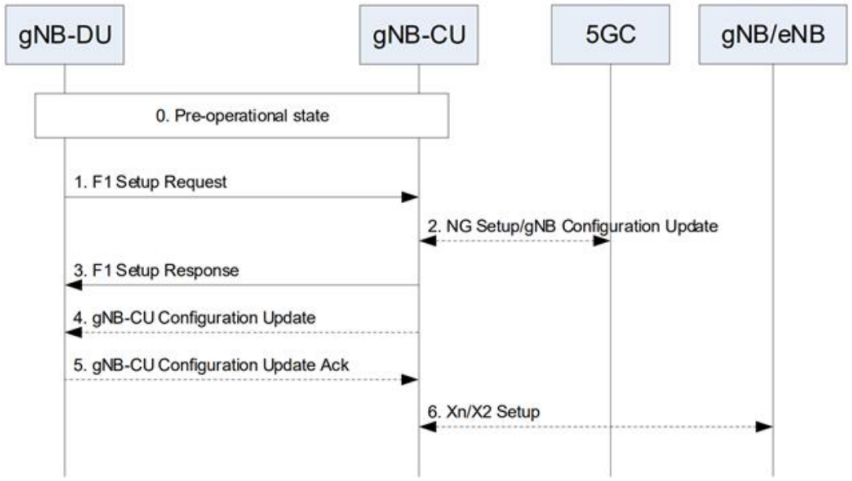

Figure 3-57: F1 interface startup and cell activation process [38.401].

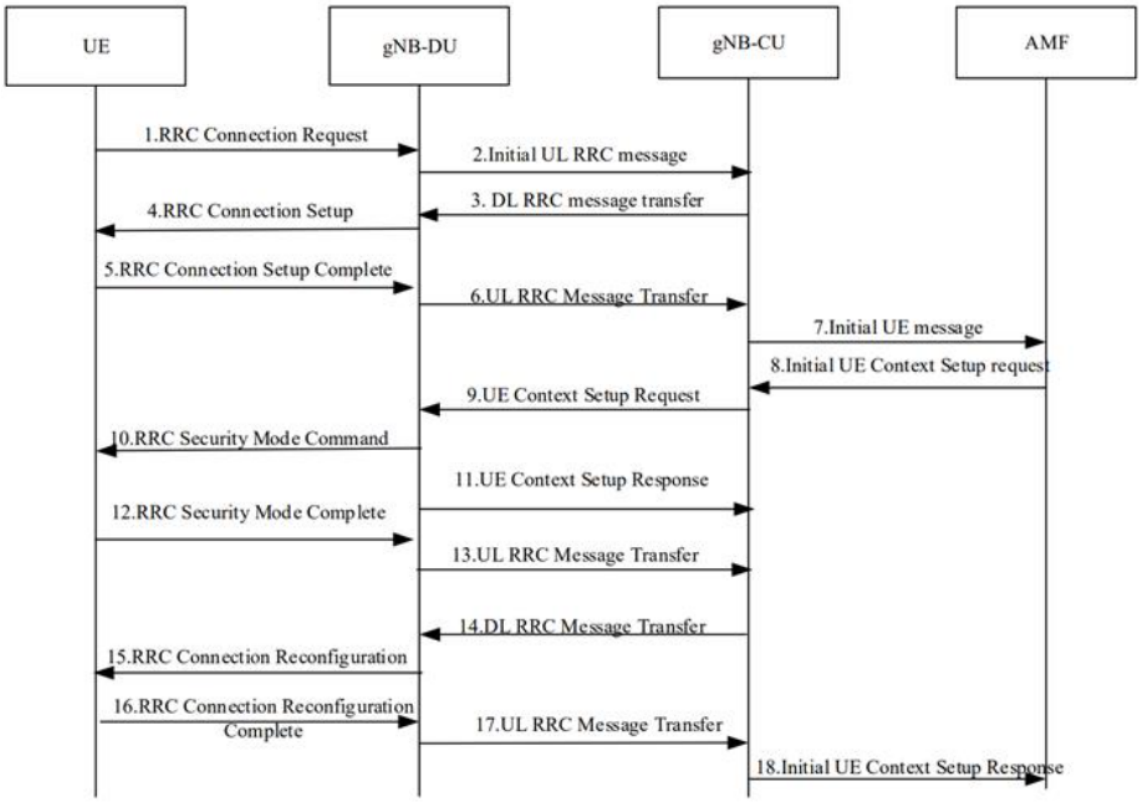

Figure 3-58: UE initial access procedure when using the DU/CU split approach [38.401].

3.6.2 Simulation Results

Here will be described the performed simulations and obtained results to compare the latency of split RAN vs. monolithic RAN. At first, the OAI 5G virtual RAN implementation has been studied to discover the functions

which perform the procedures illustrated in Figure 3-57 and Figure 3-58. Then the latency of the events of interest has been captured by implementing timer functions to relevant parts of the simulation code. Latency results are reported in below subsections to compare latencies of the split RAN vs. monolithic architecture approaches.

In both cases the used simulation environment and radio configuration parameters are as show in Table 3-2.

Table 3-2. Simulation environment characteristics and radio configuration parameters.

Hardware	Intel i7-11850H @2.5G, 16 Core, 32 GB RAM
Operating System	Ubuntu 20.0.6 LTS
Number of TX antennas	2
Number of RX antennas	2
Frequency Band	n78, ~3500MHz
SCS	30 kHz
Bandwidth	40 MHz
Duplexing	TDD
Channel Model	AWGN
Simulation Mode	Standalone NR 5G

In the simulations, instead of using a real RU, the RF simulator [OAI-RF] provided by OAI was used. RF simulator mimics functionalities of a "real" RF board but instead of sending samples over the air, it transmits them between the UE and gNB/DU by using TCP/IP interface. In addition to transfer of samples, the channel model effect can be added with RF simulator. In this study, very good (error-free) additive white Gaussian noise (AWGN) channel conditions were simulated since the focus is on the evaluation of F1 interface delays.

Note that these simulations mimic a real-time network operation, and the following result figures are illustrating RAN delay performance during a single simulation run, for each study case. However, in order to acquire confident results, multiple simulation runs are performed for each case, to check that results are approximately same for each run.

3.6.2.1 Disaggregated RAN case

In this case the simulation architecture is Alternative 2 illustrated in Figure 3-57, i.e., there is a split between DU and CU. In the OAI simulator, these components are running in own processes such that DU, CU and UE are running as separate Linux applications, Core Network components are running as virtualized docker containers. Each component is on a separate IP address, used for all its interfaces. In this all-in-one device simulation, the local loopback interface is used for each component and F1 interface is used to exchange data between DU and CU.

The high-level simulation procedure for disaggregated RAN is as follows:

1. Start CU process
2. Start DU process which triggers F1 startup (see Figure 3-57)
3. Start UE process for initial access (see Figure 3-58)
4. Collect measured processing times of the message exchange.

When Openairinterface5g CU is started, configurations are set and several threads are created such as gNB Task, RRC Task, SCTP Task, NGAP Task, CU F1 Task and GPT-U Task. After these steps, the main thread of CU waits for connection to the Openairinterface5g DU in “Entering ITTI (InTer Task Interface) signals handler” case. This is related to F1 Setup Procedure between CU and DU (see Figure 3-57). CU waits for F1 Setup Request to be sent from DU. The ITTI used here is the middleware that is used as an intra-process communication system through asynchronous message passing. Each thread in the CU structure is an ITTI task woken up or triggered by events like messages, I/O events, and timer events. The time spent on each procedure was measured, considering the introduced operation structure.

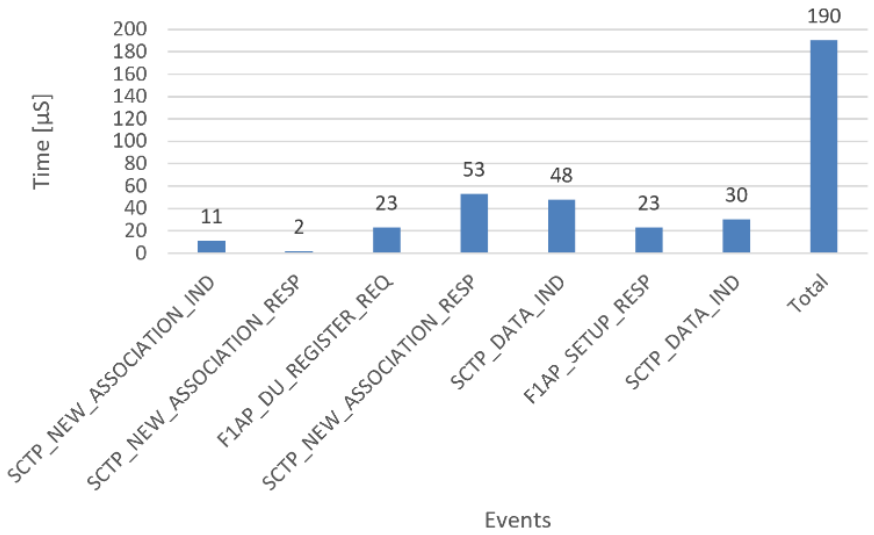

Figure 3-59: Time durations of the events required for establishment of F1 interface between DU and CU.

As in the CU case, once the DU is started, its configurations are set, and threads are created for main DU tasks. One of these main tasks, "gNB app", sends a first message, *F1AP_SETUP_REQ,* to DU task. Using this message, the unique DU task (Linux Thread) creates a DU instance context memory space, calls SCTP task to create a socket to the CU. When it receives from SCTP task the socket creation success, the DU task encodes and sends the F1 setup message to the CU. The DU initiates the procedure by sending a *F1 SETUP REQUEST* message including the appropriate data to the CU. The CU responds with a *F1 SETUP RESPONSE* message including the appropriate data. The incoming messages to the CU or DU are called "SCTP_DATA_IND". During that period, the message is decoded to discover the message type and content. Once the procedure is finished, the F1 interface is operational and other F1 messages may be exchanged between the DU and CU. Below Figure 3-59 shows the simulation results of processing times required until the F1 interface is operational, total latency being 190 microseconds.

After the F1 interface start-up, the UE initial access process will be started by using RRC message procedure, which is illustrated in Figure 5 and is explained next. At first, DU sends F1AP_INITIAL_UL_RRC_Message to CU. CU decodes F1AP_INITIAL_UL_RRC_MESSAGE that came as SCTP_DATA_IND and encodes the F1AP_DL_RRC_MESSAGE message in response and sends it to DU. According to a set of function pointers and the F1AP standard identifier "procedureCode", DU or CU calls the appropriate function. The sent and received messages descriptions for UE initial access procedure are shown in Table 3-3.

Table 3-3. Sent and received messages for UE initial access procedure. Red color-coded text illustrates the corresponding messages shown in Figure 3-58.

Description	Node	Message
First message from UE, follows up with RRC Setup or Re-establishment. UE Activating RA Procedure, Generating Random Access Response (Msg-2), RRC Setup Request (Msg-3) received, Scheduling RRC Setup.	DU	F1AP_INITIAL_UL_RRC_MESSAGE
RCC handle initial UL RRC message	CU	SCTP_DATA_IND
RRC generates RRC Setup	CU	F1AP_DL_RRC_MESSAGE
Decoding and handing DL RRC	DU	SCTP_DATA_IND
Received Ack of RRC Setup (Msg-4) CBRA procedure succeeded	DU	F1AP_UL_RRC_MESSAGE
RCC decode and handle UL RRC message (DCCH)	CU	SCTP_DATA_IND
RRC Setup Connection Completed; UE processing NR RRC Connection Setup Complete UE State = RRC Connected; Sending DL RRC Message	CU	F1AP_DL_RRC_MESSAGE
Decoding and handing DL RRC	DU	SCTP_DATA_IND

Sending UL message	DU	F1AP_UL_RRC_MESSAGE
RCC decode and handle initial UL RRC message (DCCH)	CU	SCTP_DATA_IND
RRC sending DL message	CU	F1AP_DL_RRC_MESSAGE
Decoding and handing DL RRC	DU	SCTP_DATA_IND
Sending UL message	DU	F1AP_UL_RRC_MESSAGE
RCC decode and handle UL RRC message (DCCH)	CU	SCTP_DATA_IND
UE 1 Logical Channel DL-DCCH, Generate Security Mode Command; RRC sending DL message	CU	F1AP_DL_RRC_MESSAGE
Decoding and handing DL RRC	DU	SCTP_DATA_IND
Sending UL message	DU	F1AP_UL_RRC_MESSAGE
RCC handle UL RRC message	CU	SCTP_DATA_IND
Received Security Mode Complete; UE 1 Logical Channel DL-DCCH; Generate UE Capability Enquiry RRC sending DL message	CU	F1AP_DL_RRC_MESSAGE
Decoding and handing DL RRC	DU	SCTP_DATA_IND
Sending UL message	DU	F1AP_UL_RRC_MESSAGE
RCC decode and handle UL RRC message (DCCH)	CU	SCTP_DATA_IND
UE 1 Received UE Capabilities, send message to NGAP: NGAP_UE_CAPABILITIES	CU	F1AP_DL_RRC_MESSAGE
Decoding and handing DL RRC	DU	SCTP_DATA_IND
Sending UL message	DU	F1AP_UL_RRC_MESSAGE
Sending UL message	DU	F1AP_UL_RRC_MESSAGE
RCC decode and handle UL RRC message (DCCH)	CU	SCTP_DATA_IND
RCC decode and handle UL RRC message (DCCH)	CU	SCTP_DATA_IND
PDU Session Setup Initiating message; UE 1 configure DRB ID 1 for PDU session ID 10. Selecting CU-UP ID; UE 1 associating to CU-UP; GTP-U created tunnel for UE ID 1; UE 1 trigger UE context setup request with 1 DRB	CU	F1AP_UE_CONTEXT_SETUP_REQ
Decoding and handing DL RRC	DU	SCTP_DATA_IND
Created tunnel for UE (NGAP)	DU	F1AP_UE_CONTEXT_SETUP_RESP
RCC decode and handle UL RRC message (DCCH)	CU	SCTP_DATA_IND
UE 1 updating PDU session ID; UE 1: Generate RRC Reconfiguration; UE 1: PDU Seesion ID modified; E1 Updating PDU Session ID	CU	F1AP_DL_RRC_MESSAGE
Decoding and handing DL RRC	DU	SCTP_DATA_IND
Sending UL message	DU	F1AP_UL_RRC_MESSAGE
RCC decode and handle UL RRC message (DCCH); UE 1 Receive RRC Reconfiguration; NGAP_PDUSESSION_SETUP_RESP: sending message	CU	SCTP_DATA_IND

Figure 3-60 shows the time duration results for the UE initial access procedure based on the events explained in Table 3-3. In the simulation results shown in Figure 3-59 for the processes of Figure 3-58, only the time it takes from the moment the message reaches the DU to process it, encode the necessary message and send the response to the CU, and the time it takes for the CU to receive the relevant message, decode it, process it, encode the relevant response and send it to the DU are taken into account. The total time duration for UE initial access is 1426 microseconds for the splitted DU/CU case, based on the simulation results show in Figure 3-60.

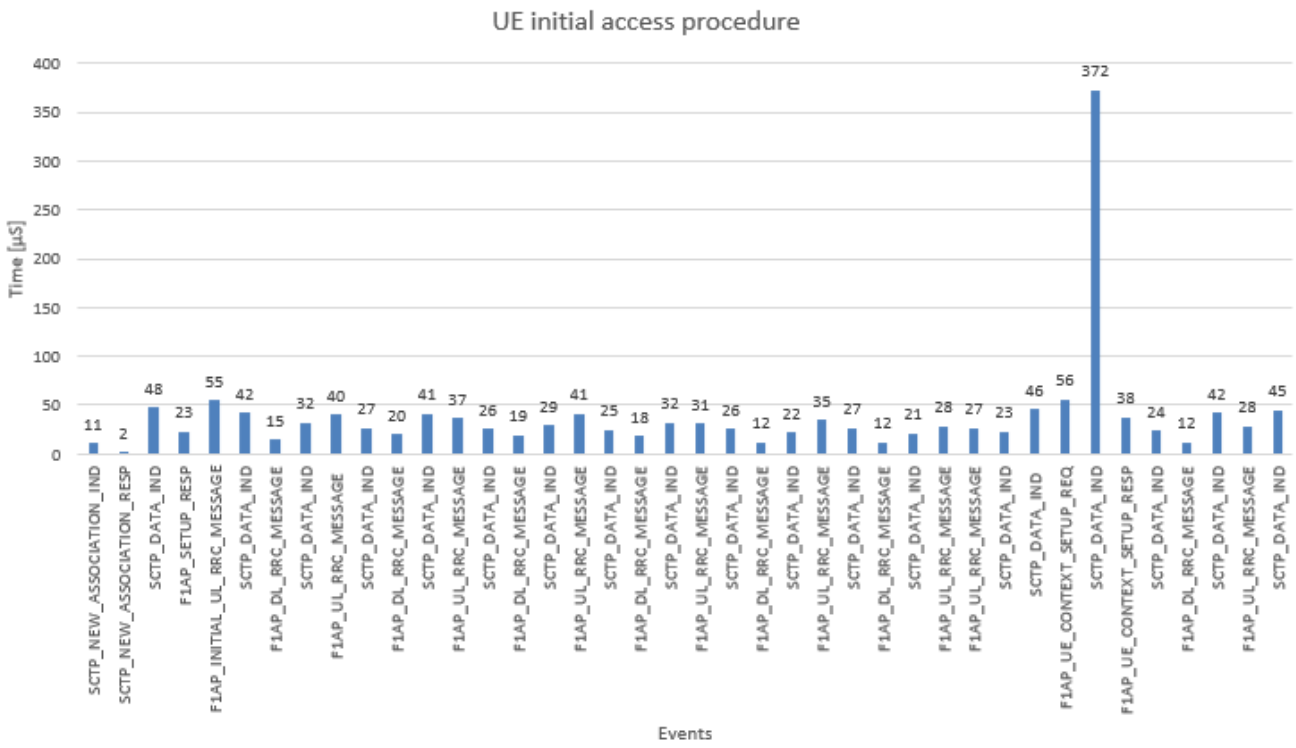

Figure 3-60: Time durations of events at DU and CU during UE initial access procedure.

3.6.2.2 Monolithic RAN case

In this case the simulation architecture is Alternative 1 illustrated in Figure 3-56, i.e., there is not a split between DU and CU. Therefore, the F1 interface startup is not needed to exchange data between DU and CU, and the focus here is on the simulation of UE initial access procedure.

The high-level simulation procedure for monolithic RAN is as follows:

1. Start monolithic gNB process
2. Start UE process for initial access (see Figure 3-58)
5. Collect measured processing times of the message exchange

When gNB and UE are started, first the configurations are set, then the threads of the main processes are created. In order to establish communication via RF simulator, gNB first waits for UE Client requests to come via TCP/IP socket. After the TCP/IP connection is established, the initial access process is started. The message flow is the same in both F1 and monolithic cases, with the difference that F1AP encodes the messages using ASN.1 and sends over a socket. This means that, unlike the F1 split case, there are no SCTP_DATA_IND messages.

In the monolithic architecture case, there is a direct communication between MAC and RCC. Since no decoding/encoding process is required, the data coming from RRC can be immediately processed in the MAC and the relevant response can be sent. For this reason, the UL in the message flow was ignored except F1AP_UE_CONTEXT_SETUP_RESPONSE. The sent and received messages for UE initial access for monolithic case are introduced in the Table 3-4.

Table 3-4. Monolithic gNB MAC-RRC message flow. Red color-coded text illustrates the corresponding messages shown in Figure 3-58.

Description	Message
First message from UE, follows up with RRC Setup or Re-establishment.	F1AP_INITIAL_UL_RRC_MESSAGE
Activate SRB 1, RRC generate RRC Setup	DL_RRC_MESSAGE_Transfer
UE 1 Processing NR RRC Setup Complete; UE State = RRC Connected; RRC sending DL Message	DL_RRC_MESSAGE_Transfer

RRC sending DL Message	DL_RRC_MESSAGE_Transfer
Selected security algorithms; UE 1 Logical Channel DL-DCCH; Generate Security Mode Command	DL_RRC_MESSAGE_Transfer
Received Security Mode Complete; UE 1 Logical Channel DL-DCCH; Generate NR UE Capability Enquiry	DL_RRC_MESSAGE_Transfer
UE 1 Received UE Capabilities; Send message to NGAP: NGAP_UE_CAPABILITIES	DL_RRC_MESSAGE_Transfer
PDU Session Setup Initiating message; UE 1 configure DRB ID 1 for PDU session ID 10; Selecting CU-UP ID; UE 1 associating to CU-UP; GTPU created tunnel for UE ID 1; UE 1 trigger UE context setup request with 1 DRB	F1AP_UE_CONTEXT_SETUP_REQUEST
Received NGAP Initial Context Setup Response	F1AP_UE_CONTEXT_SETUP_RESPONSE
UE 1 updating PDU session ID; UE 1 Generate RRC Reconfiguration	DL_RRC_MESSAGE_Transfer

Simulation results for the monolithic RAN case are shown in Figure 3-61. Based on these results, the total time duration for UE initial access in monolithic case is 339 microseconds.

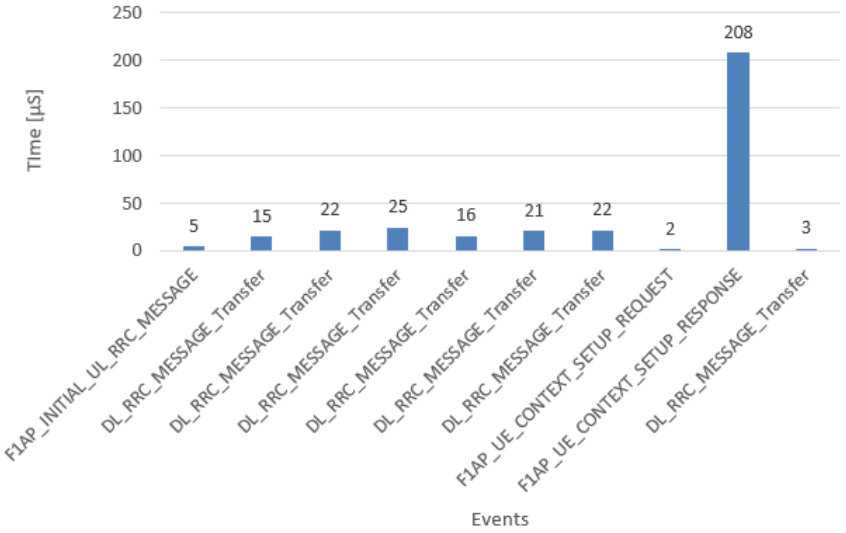

Figure 3-61: Time durations for events occurred during UE initial access procedure in monolithic RAN case.

3.6.3 Discussion

As can be seen from the simulation results of processing times of the messages, the disaggregated RAN has substantially bigger latency in this UE initialization case since it requires the F1 interface establishment and message exchange between DU and CU. F1 interface establishment must be performed when DU and CU are started to enable their communication with each other.

Figure 3-62 shows as a summary, the total time durations for disaggregated RAN (DU and CU) and monolithic gNB case, based on the simulation results introduced above. F1 interface startup total time is shown separately since it is not needed if the DU and CU are already connected an interface has been established for their message exchange purposes. As can be seen from the results, monolithic case has substantially lower latency for UE initial access process since it does not require message exchange through F1 interface between DU and CU. Note that these results do not include the physical distance between RAN components because in the simulations they are running in the same machine. Physical distance would further increase the delay in the disaggregated RAN case since the signal propagation delay should be taken into account.

The main reason for the much lower latency in monolithic case is that SCTP_DATA_IND messages processing is not needed in the monolithic architecture case. Total time used only for SCTP_DATA_IND messages is 902 µs in the splitted DU and CU case. Note that this total time used for decoding messages in CU is not needed in the monolithic gNB case.

In further work, the simulations will be performed to study other scenarios, e.g., handover required due to UE movement or varying channel conditions, larger amount of UEs and DUs connected to single CU and RAN architecture design selection affect to D-MIMO implementation complexity and communication performance

optimization. HLS split has a crucial role in all those scenarios since RRC is involved, and UEs' and channel status information are need for optimal and low latency RAN control decisions. Many 6G use cases such as controlling of the industrial robots/cobots and autonomous vehicles requires ultra-reliable and ultra-low latency communication, which development can be supported by the virtual performance evaluation already during the radio enabler design phase. The presented simulation study will be used as a basis for further work, which results will be reported in the upcoming deliverable of this project.

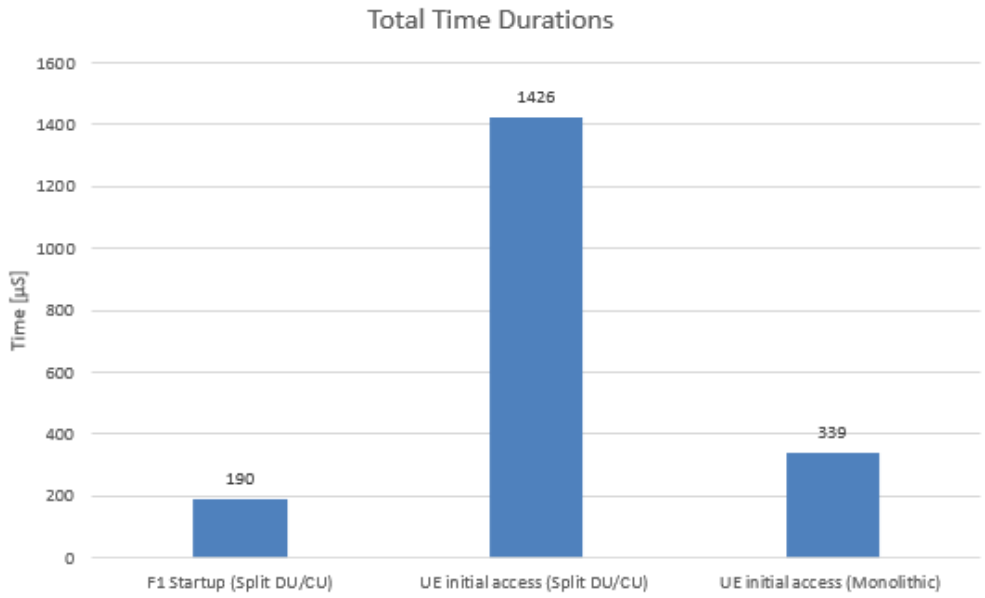

Figure 3-62: Simulated total times for split DU and CU, and monolithic gNB.

4 Evaluation of the security, privacy and system-level resilience

This chapter reports the initial evaluation results of the different Security, Privacy and Resilience (SPR) enablers identified in [HEX223-D22] and [HEX224-D23], addressing the goals of assuring a trustworthy environment and supporting resilience and availability, as described in section 2.2 for principles 5 and 6. The reporting is structured according to the TRL of each of these controls. Together with these results, the integration patterns already identified and analysed in conjunction with other WPs, for some of those enablers will be discussed.

It is worth noting this document reports a series of intermediate results, ranging from the definition of evaluation mechanisms to any relevant measure obtained, when available. These intermediate results constitute the foundation for the final results and integration patterns to be reported in the final E2E system design. Given these circumstances, not all enablers identified in [HEX224-D23] are reported here. In particular, the specific definition of the Level of Trust Assessment Function (LoTAF) is part of a joint effort with T2.3, which is not reported in this deliverable but will be extended in the forthcoming D6.5, and the security orchestration mechanisms are still being refined within the ACROSS project [ACR24] where they originated.

4.1 Enablers at Basic TRLs

These enablers are essentially associated with Physical Layer Security (PLS) features, and this section provides the outcomes of the conceptual evaluation, by means of simulation or experimental laboratory environments, of the related SPR enablers, together with any progress on the concepts themselves, as further theoretical and experimental evidence is gathered.

4.1.1 Physical Context Awareness

Context-awareness enabler in wireless systems utilizes the dynamic characteristics of the environment, such as physical obstacles, movement, and channel conditions, to generate unique keys based on the current state of the wireless channel. This adaptability impacts the Key Generation Rate (KGR) through various factors like environment type, obstacles, movement, and other physical parameters. Recognizing the condition of the wireless channel allows the system to adjust the KGR appropriately, as demonstrated by analyzing two implementations that highlight key physical parameters in context-aware scenarios.

Implementation 1: Dataset [WKS21] includes ultrawideband channel state information and terminal location data collected using consumer-grade hardware. The setup utilizes Deca Wave EVB1000 boards in an indoor office environment, with a mobile tag on a self-driving robot to facilitate physical layer key derivation and simulate various scenarios. The measurement environment is a typical office room, 5.9 m long and 3.05 m wide, with four anchors (A0, A1, A1', A2) attached to the walls at a height of 1150 mm. Four EVB1000 boards are used, with four fixed as anchors and one acting as the mobile tag. In this setup, the roles are defined as follows: the mobile tag is Bob, A1 and A1' represent Alice, and A0 and A2 are designated as Eves 1 and 2, respectively. In the symmetric scenarios, A0 (Eve 1) and A1 (Alice) are symmetrically positioned relative to the walls, A2, and the corners, which can influence signal reflections. In contrast, in the asymmetric scenarios, A0 (Eve 1) and A1' (Alice) are asymmetrically positioned relative to the walls, A2, and the corners. The following scenarios are considered in implementation 1.

1) Asymmetric - Constant Speed: Bob moves steadily at 0.15 m/s compared to Alice, while their positions relative to the anchors and walls are asymmetric.
2) Asymmetric - Static: Alice and Bob stay still, positioned asymmetrically in relation to the anchors and walls.
3) Symmetric - Varying Speed: Bob begins moving at 0.25 m/s, progressively decelerating to 0.05 m/s, while their positions relative to the anchors and walls are symmetric.

Implementation 2: The setup [HEX2-MMC+23] includes three Universal Software Radio Peripheral (USRP 310s) which were configured as Alice, Bob and Eve. Alice and Bob are the legitimate users and Eve is the eavesdropper. Using Time Division Duplexing (TDD) scheme, Alice and Bob exchanged signals with each other while Eve recorded the received signals at her end simultaneously. The chirp signals were transmitted at

3.75 GHz (i.e., 8 cm wavelength) at a bandwidth of 160 MHz. Alice and Bob were at fixed positions (4 m apart) while recording Eve's measurements at different locations and the distances are multiples of the wavelength. Non-Line of Sight (NLoS) was introduced by placing absorbers in the direct path. Static measurements were done during nighttime. Movements of people and objects contributed to dynamic measurements. The following scenarios are considered in Implementation 2.

1) Line of Sight (LoS) – Dynamic: Alice and Bob are moving and there is LoS between them.
2) Line of Sight – Static: Alice and Bob are static and there is LoS between them.
3) Non-Line of Sight – Dynamic: Alice and Bob are moving and there is not LoS between them.
4) Non-Line of Sight – Static: Alice and Bob are static and there is not LoS between them.

Classification Using Feature Extraction: Machine learning was utilized to classify environments based on Channel Impulse Responses (CIRs), necessitating the extraction of two types of features: momentary features, capturing properties of individual CIRs, and temporal changeable features, representing changes across consecutive CIRs over time. Momentary features included absolute and phase differences between samples, attributes of LoS components such as amplitude and phase of the statistical measures like average, variance, kurtosis, and skewness. Temporal changeable features encompassed cross-correlation between consecutive CIRs, rates of change in LoS component attributes, and Principal Component Analysis (PCA). These features were used to label windows of CIRs for evaluation with various classification methods, including Logistic Regression, Decision Trees, Random Forest, Support Vector Machines (SVM), k-Nearest Neighbours (kNN), and Convolutional Neural Networks (CNNs). Finally, Random Forest emerged as the optimal classifier, with selected features, including cross-correlation, sample differences, component variance, LoS amplitude, and phase change rate proving most effective for environment classification.

Evaluation Results: The confusion matrices for two implementations are shown in Figure 4-1 and Figure 4-2, respectively. Table 4-1 and Table 4-2 represent the classification tables for both the implementations. The results in both the implementations indicate a near-perfect performance in scenario classification. Implementation 2 achieved a flawless accuracy of 100%, demonstrating its exceptional ability to identify scenarios accurately. Conversely, implementation 1 achieved an overall accuracy of 87%, signifying robust performance with slight room for improvement. Notably, the minor discrepancies mainly arose between scenarios featuring variable and constant speeds. This occurred due to the overlap in speed ranges between the two scenarios, posing a challenge for precise differentiation.

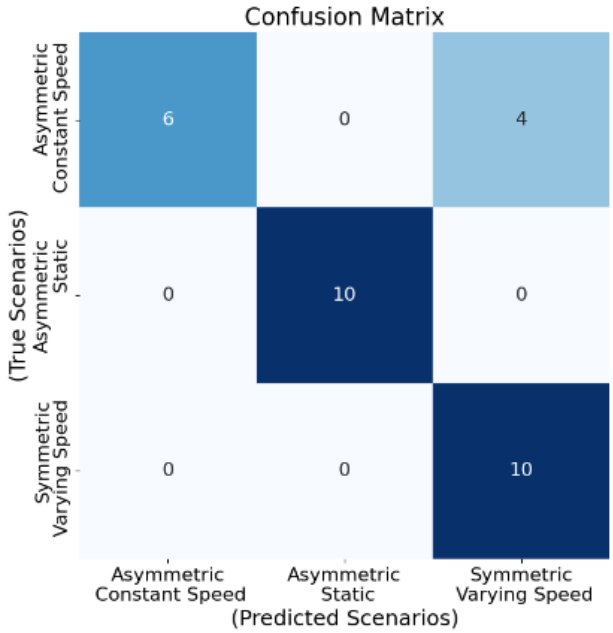

Figure 4-1: The confusion matrix for Implementation 1.

Table 4-1: Classification table for Implementation 1.

Accuracy = 0.867	Precision	Recall	f1-score	Support
Asymmetric-Constant Speed	1.00	0.60	0.75	10
Asymmetric-Static	1.00	1.00	1.00	10
Symmetric-Varying Speed	0.71	1.00	0.83	10

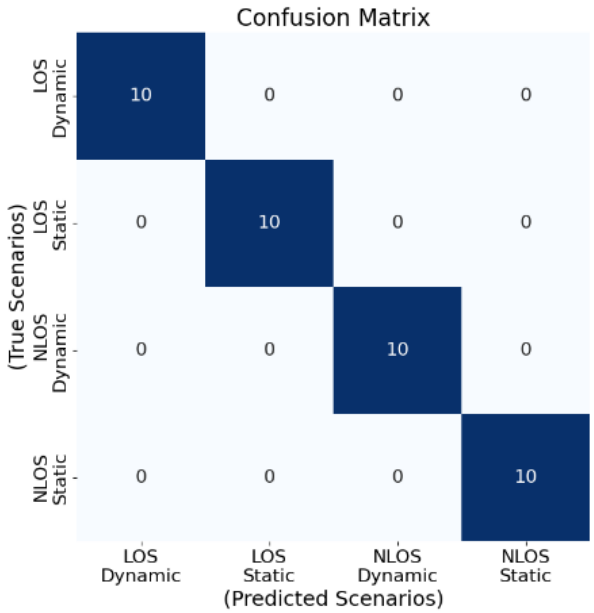

Figure 4-2: The confusion matrix for Implementation 2.

Table 4-2: Classification table for Implementation 2.

Accuracy = 1.0	Precision	Recall	f1-score	Support
LOS-Dynamic	1.00	1.00	1.00	10
LOS-Static	1.00	1.00	1.00	10
NLOS-Dynamic	1.00	1.00	1.00	10
NLOS-Static	1.00	1.00	1.00	10

4.1.2 Physical Anomaly Detection

Two SPR controls have been proposed in support of this enabler, and the initial results regarding them are reported below.

4.1.2.1 Anomaly Detection in Disaggregated RAN

This section provides the outcomes of FL-based anomaly detection algorithm presented for disaggregated RAN as introduced in [HEX224-D23]. The federated client models within disaggregated RAN nodes are trained using locally available data, and model vectors are communicated to the relevant first-level aggregators hosted in RAN intelligent controllers (RICs). In this experiment set up, the detection of anomalies is performed by detecting malicious UEs that are distributed across the network and to identify the slice type based on the performance indicators generated at the gNB per UE [HEX2-RAP+24].

For the normal traffic class, real-world 5G traces were collected in a variety of conditions for each traffic slice (i.e., enhanced mobile broadband (eMBB), massive machine type communication (mMTC), and ultra-reliable low-latency communication (URLLC).) These traces are generated by the Colosseum emulator, an open RAN emulator designed for real-world network traffic generation [BJP+21]. The performance indicators are reported per UE basis every 250 ms to create a robust dataset for the normal traffic class that represents a wide range of real user traffic patterns.

Two attack models are considered for generating anomalous traffic class. The first model focuses on a User Datagram Protocol (UDP) Distributed Denial of Service (DDoS) attack, where an attacker-UE inundates the gNB with a substantial volume of UDP packets, thereby degrading system performance. The second model introduces a more sophisticated attack variant known as the bandwidth hog attack. This attack represents a attempt to disguise a DDoS attack by closely mimicking realistic packet arrival rates.

The goal of this experiment is to detect malicious UEs which are distributed across the network and to identify the slice type based on the performance indicators generated at the gNB per UE. These performance indicators are taken as follows: for downlink number of samples transmitted since last time stamp, current queue length in bytes, throughput in Mbps, number of packets transmitted since last timestamp; for uplink number of samples received since last time stamp, current queue length in bytes, throughput in Mbps, number of packets received since last time stamp. The accurate classification of normal traffic slices to eMBB, URLLC, and mMTC required both uplink and downlink features. To analyze the traffic flows, a Long Short-Term Memory (LSTM)-based classification model is used. The main motivation is that LSTM is designed to avoid long and complex dependency issues and can remember extensive historical information [LL19]. The model consists of one LSTM layer and one fully connected layer followed by a softmax layer (Figure 4-3).

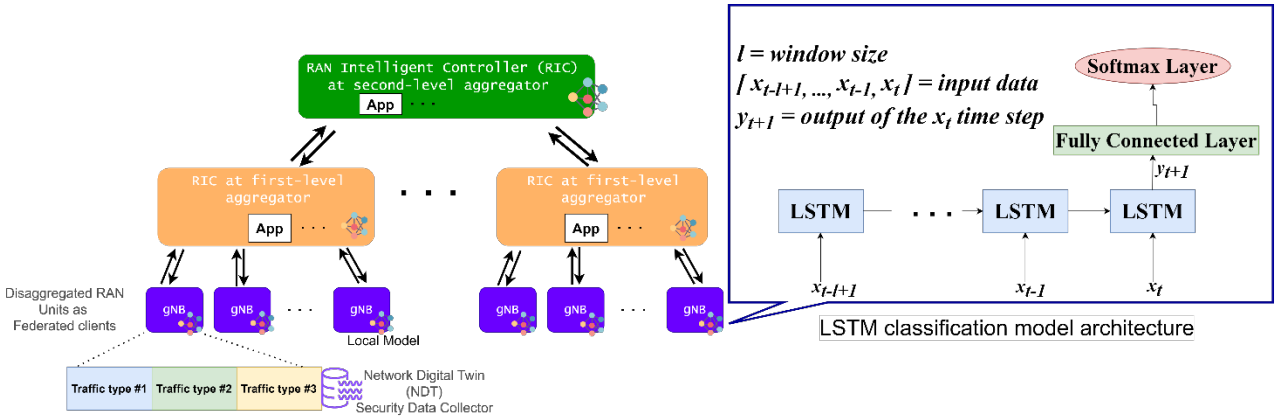

Figure 4-3: Logical experimental setup considered for FL-based anomaly detection in disaggregated RAN architecture and LSTM classification model.

Each client is considered to have one normal traffic slice (e.g., eMBB, mMTC, URLLC) and at least one anomalous traffic slice. The data pre-processing involves two main steps: data scaling and data windowing. Standardization was used for data scaling, and data windowing was applied to structure the data sequentially, making it suitable as input for the LSTM model. Due to the temporal dependencies in the data, the time series data was partitioned into training, testing, and validation sets based on sequential time values. The first 80% of the time-series records were allocated to the training set, and the subsequent 20% were designated as the testing set. For model validation during training rounds, the last 20% of time records from the training set were extracted and defined as the validation set.

Performance results for the trained LSTM classification model are provided in Table 4-3. Model performance increases as the window size is reduced. As such a noticeable accuracy enhancement in traffic classification was observed with the reduction in window size, indicating shorter temporal dependencies in the dataset. Consequently, the model can make real-time predictions based on this dataset. This emphasizes the realization of FL based model training in such a NDT environment, which can be used to tailor the model's configuration to the specific characteristics of the data.

Table 4-3: Performance results for LSTM classification model with different window sizes in time scale of seconds.

Window size (s)	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
1	99.87	99.87	99.83	99.85
4	99.73	99.77	99.65	99.71
10	99.6	99.6	99.49	99.54
50	96.65	94.21	95.35	94.44
100	95.34	93.04	93.48	92.0

4.1.2.2 DT-Enabled Spectrum Anomaly Detection

Employing a DT of the radio environment to exploit contextual awareness for spectrum anomaly detection was proposed in [HEX224-D23]. Spectrum anomaly hereby refers to any type of interfering signal, no matter whether its interference is intentional (jamming) or unintentional (faulty devices or non-compliance to regulations). The approach is targeting on licensed bands where the legitimate transmitters (TXs) are known by the network. The architectural details can be found in [HEX224-D23].

To evaluate the approach, ray tracing simulations were executed using Sionna [HCA+22]. The evaluation is extensively described in [KBS+24], therefore we provide only a summary here. The scenario is oriented on a small factory hall visualized in Figure 4-4, with a dimension of 40m x 40m limited by concrete walls, and production lines simplified as metal boxes. As the current state of work is purely simulation-based, in a first step the mimicked "real-world" data are generated, we refer to this as physical twin (PT) in the following. In a normal state, legitimate TXs with isotropic antennas are placed at random locations. Next, the path loss between each legitimate TX and each sensing unit (SU) is derived from ray tracing. It is assumed, that the SUs monitor the wideband received signal strength (RSS) of the band in which the previously mentioned TXs are active. Thus, the RSS is calculated by adding up the transmit power of each TX reduced by the respective path loss. To model the DT, the same simulation is run with random location offsets of the TXs which represent the localization error. The vector of differences $\boldsymbol{\Delta}$ between the PT and the DT RSS values, defined by its entries $\Delta_j = P_{\mathrm{rx},j} - \hat{P}_{\mathrm{rx},j} + \varepsilon, j \in \{1, \ldots, N_{su}\}$ for each SU j serves as input for the anomaly detection approach. Hereby, ε is a random variable which reflects further deviations between the PT and the DT such as electronic noise and inaccuracy of the ray tracing model.

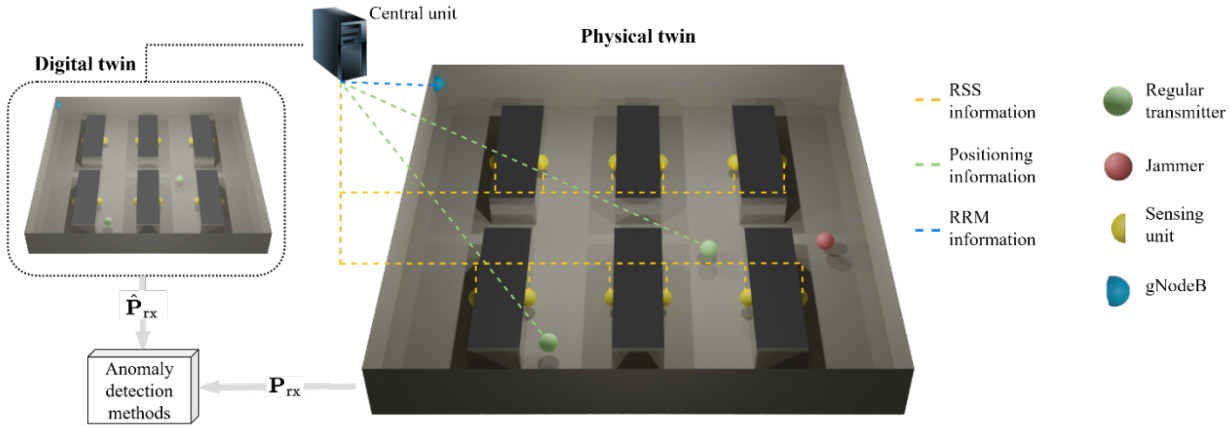

Figure 4-4: Scenario and system architecture for the DT-enabled spectrum anomaly detection [KBS+24].

An unsupervised approach is followed, i.e., the statistics of $\boldsymbol{\Delta}$ from the normal state are generalized, for which different algorithms are compared. In a next step, further normal samples are created, whereby each sample is represented by one vector $\boldsymbol{\Delta}$. Moreover, anomaly samples are created by placing an additional transmitter, i.e., the jammer, at a random position, which is not reflected in the DT but only in the PT and thereby increases the deviation between the PT and DT.

The generated collection of normal and anomaly samples is subsequently used to test the approach. Two algorithms are compared for generalizing the statistics of normal data, an empirical energy detector (ED) and

the unsupervised learning algorithm local outlier factor (LOF). The receiver operating characteristics (ROC) are shown in Figure 4-5. The error term ε is modelled as zero mean, normally distributed with different standard deviations, indicated by the colour of the line.

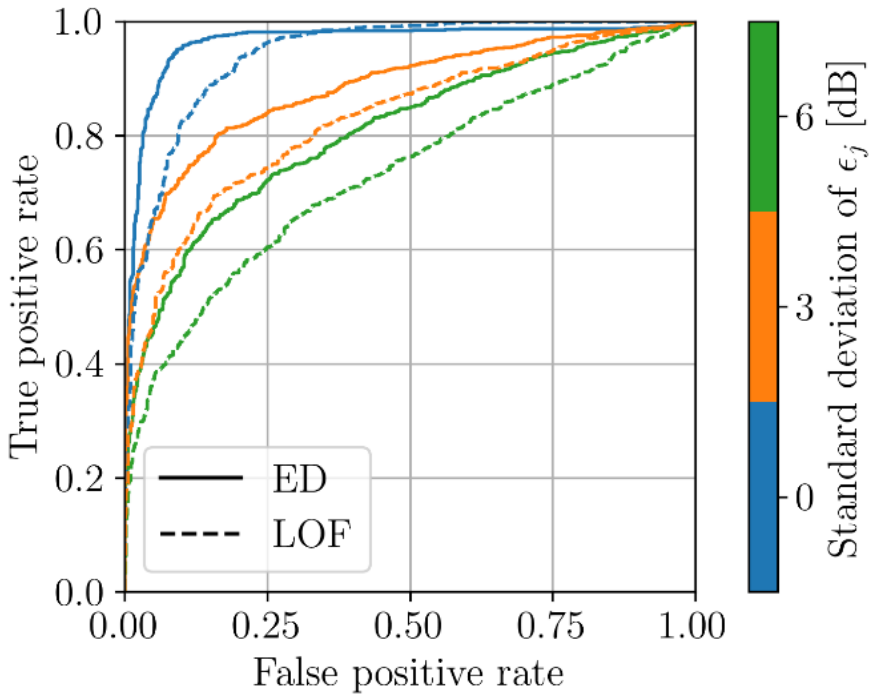

Figure 4-5: ROC curves for spectrum anomaly detection using a DT of the radio environment.

It can be seen in the figure, that the approach achieves a strong spectrum anomaly detection performance in case, the DT model accurately reflects the PT, i.e., the standard deviation of ε is not too big, whereby the specific value depends on the specified accuracy requirements. Moreover, the best performance can be achieved when using the ED instead of the LOF.

4.1.3 JCAS Threat Mitigation

Joint Communication and Sensing (JCAS) introduces novel network functions, similar to Sensing Processing Functions (SPF) and Sensing Control Function (SCF), into the core network as the Sensing Management Functions (SeMF) [WGM+23]. The introduction of these additional sensing functions into the core network poses several security and privacy threats. Firstly, it expands the attack surface of the network, providing more avenues (interfaces, data storage, integration points, etc.) for malicious actors to exploit vulnerabilities. The attackers could exploit vulnerabilities in one sensing function to breach trust boundaries and gain unauthorized access to other network components. Secondly, the sensing data collected through these sensing functions heightens the risk of data breaches and privacy violations.

Considering the sensitivity of the sensing data and the privacy of sensing targets, architectural enhancements have been introduced to the core network. A new network function called Sensing Policy, Consent, and Transparency Management (SPCTM) has been designed [HEX2-DUN+24]. This function serves as an administration tool for enforcing security and privacy controls within the sensing ecosystem. Its primary purpose is to ensure that other network functions involved in sensing activities can adhere to privacy preservation principles effectively along with sensing and consent policies. A dedicated repository called the "Sensing Store" is also proposed to record sensing policies, Use Case (UC) data, sensing disclosure logs, consent data, and transparency data. This systematic compilation and storage of data could enhance the overall management and transparency of the sensing ecosystem, while improving compliance with General Data Protection Regulation (GDPR) and International Organization for Standardization (ISO) privacy frameworks. Figure 4-6 shows the responsibilities of the SPCTM, the information stored in the sensing store, and the interfaces for these newly introduced components.

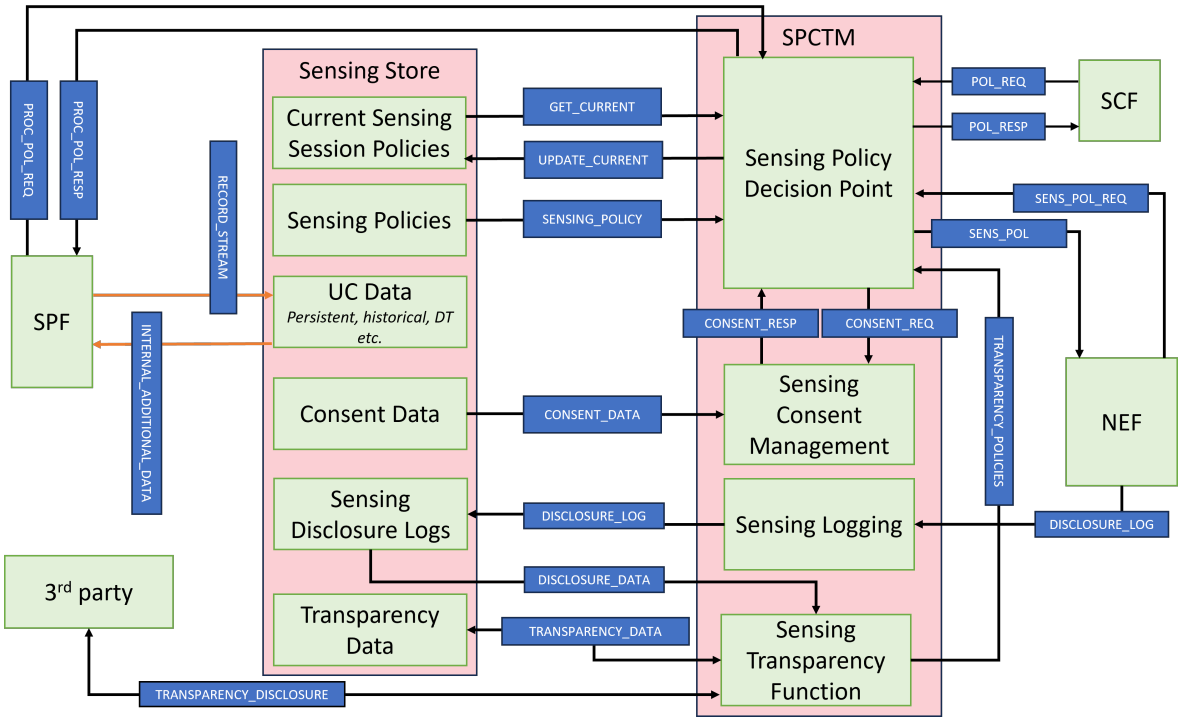

Figure 4-6: Role of SPCTM and sensing store in the JCAS architecture.

With the newly introduced interfaces and components in the architecture, the security and privacy threats associated with each interface were analyzed. The detailed discussion of the interfaces, threat analysis, and suggested mitigative controls are presented in [HEX2-DUN+24]. For threat mitigation, several security and privacy controls for the JCAS system were proposed. These include authentication controls, access and sensing information flow control, as well as information retention and disposal measures. Additionally, confidentiality and integrity controls for sensing data in storage and for signals and data in transit were recommended. Privacy controls to safeguard an entity's PII from discovery and misuse, encompassing anonymity, pseudonymity, unlinkability, and unobservability, were also suggested. It was noted that sufficient mechanisms are needed for collecting and managing consent from UE and targets (wherever applicable) and informing targets about ongoing sensing activity which concerns them, to comply with the principle of "openness, transparency, and notice". On the other hand, solutions are needed for data authentication and for the secure export and import of data to and from the JCAS system, especially when third-party interaction is involved. Additionally, it is crucial to design and implement protection mechanisms against physical layer attacks such as jamming and denial-of-service attacks. Furthermore, safeguarding synchronization mechanisms on the radio side is necessary to ensure the resilience and availability of the sensing service.

When integrating sensing units into User Equipment (UE), it is imperative to prioritize security and privacy controls. These controls serve to protect sensitive sensing data, to uphold the UE's interests in consent and policy management and to ensure privacy-preserving and consent abiding sensing in UE. To achieve this, it is suggested to implement functionalities similar to the SPCTM on the UE side. This enables the management of sensing policies, obtaining consent for enabling UE sensing services, and ensuring transparent data disclosure to the network. Furthermore, given the involvement of UE resources in processing sensing data, network functions similar to the SCF and SPF on the network side are essential for effective sensing management in the UE.

4.1.4 Physical Layer Deception

The concept of physical layer deception (PLD) was proposed in [HEX2-HZS+23], and briefly described in [HEX224-D23] in context of the Hexa-X-II security framework. Its principles are illustrated by Figure 4-7 (up). The transmitter randomly selects a ciphering key from a predefined key pool and uses it to encrypt its outgoing message with a symmetric block encryption algorithm. The encryption codec shall be carefully designed to guarantee a closure, i.e. the sets of valid ciphertext codewords and of valid plaintext codewords

must be identical. The key and the ciphertext are then individually processed through channel coding, modulation, and power adaptation, before multiplexed and transmitted together. While power-domain non-orthogonal multiplex is considered in the figure, the framework does not decline other multiplexing schemes such as orthogonal frequency domain multiplexing (OFDM). By appropriately optimize the coding rate and power mixing ratio, it is possible to selectively apply PLS only on the key component for protection from potential eavesdropping, while exposing the ciphertext component thereto. Thus, while the legitimate receiver, benefiting from its satisfactory channel condition, can likely decode both components of the message and therewith recover the plaintext, an eavesdropper with poor channel condition will not be able to decode the key but only the ciphertext. Due to the closure of the encryption codec, this ciphertext will be misunderstood by the eavesdropper as a plaintext, which leads to a deception. In case the eavesdropper is aware of this deceptive coding scheme (e.g., when the approach is publicly standardized, or due to knowledge leakage,) the ciphering can be randomly activated and deactivated. When deactivated, the plaintext is directly sent to the channel coder, and a random litter sequence is used to replace the channel-encoded key, to prevent revealing the ciphered status through the power profile of the transmitted message, as illustrated in Figure 4-7 (down).

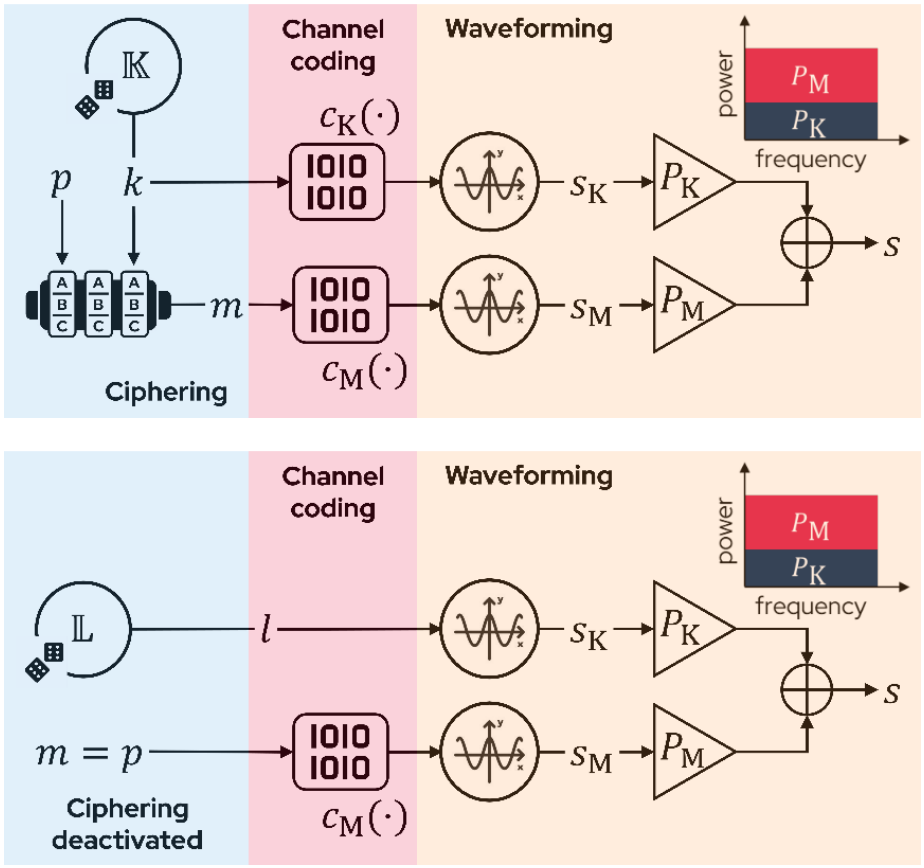

Figure 4-7: The transmitting scheme of physical layer deception [CHZ+24]. (up) Deceptive ciphering activated. (down) deceptive ciphering deactivated.

By setting an upper-bound for the leakage-failure probability (LFP), i.e., the probability that the plaintext is either correctly perceived by the eavesdropper, or not perceived by the legitimate receiver, the PLD transmitting scheme can be optimized regarding the channel coding rate of ciphering key and the power multiplexing ratio, under certain specifications of message block length, plaintext coding rate, and transmission power budget. Numerical simulation results are depicted in Figure 4-8, where two implementations of classical PLS are taken as baselines: one optimizes the multiplexing ratio under a fixed key coding rate, while the other optimizes the key coding rate under a fixed multiplexing ratio. It shows that the PLD approach can not only enable deception, but also outperform classical PLS baselines regarding the LFP with enough power budget, especially when the eavesdropping channel is good.

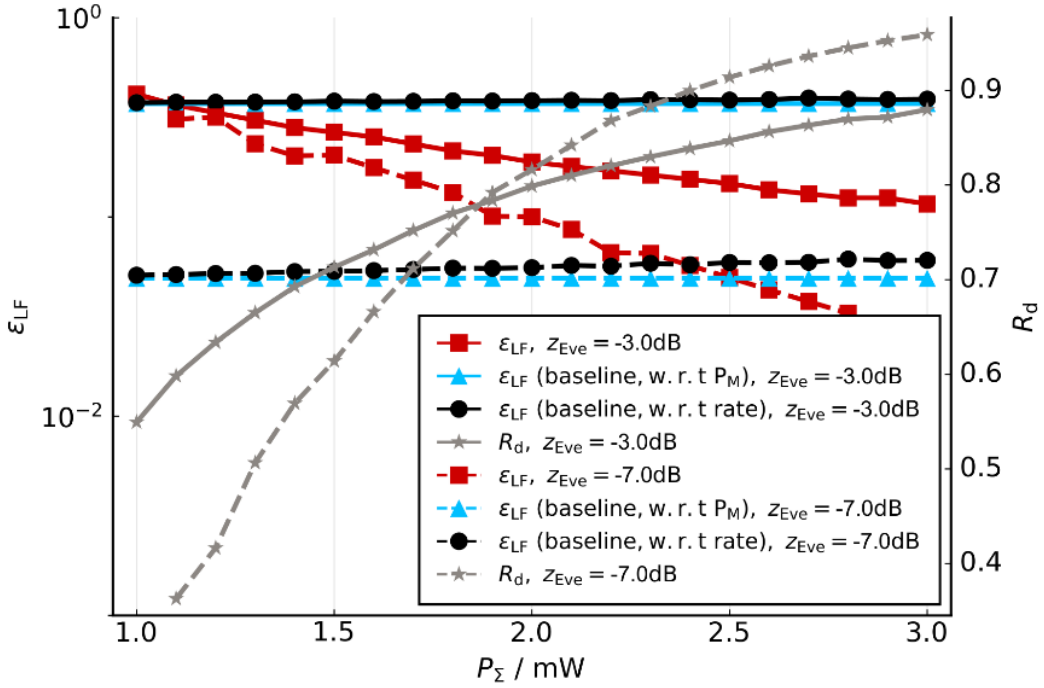

Figure 4-8: Benchmark results of PLD against classical PLS baselines [CHZ+24].

4.2 Enablers at Validation TRLs

The enablers at this stage are mostly associated with the trust fabric future network services will rely on, the use of simulation environments, and the lifecycle management of security controls. This section provides the experimental environments being devised and applied to evaluate the supporting SPR controls and the corresponding enabler interfaces. In those cases where some experiments have already been performed, and measurements relevant enough are available, these are provided and discussed, together with the foreseen characteristics of the interfaces for using these enablers or directly interacting with the corresponding controls.

4.2.1 Trustworthy AI

The development and use of AI and ML systems has become very important. Ensuring that these systems are robust, secure, and transparent is critical to their reliable operation. Given the complexity and black-box nature of ML models, protecting these systems from adversary and privacy attacks is a major challenge. These attempts can have serious consequences and disrupt networks. To solve these problems, it is important to investigate advanced methods to have more robust and privacy enhanced (i.e., trustworthy) AI/ML solutions.

This section focuses on security and privacy concerns in AI and ML systems. First, a Federated Learning (FL) approach is described, which improves data and models security and privacy. Later, it is discussed how XAI can be used to improve ML security and explain patterns and better understanding to support vulnerability and prevention.

4.2.1.1 A framework for security and privacy in federated learning

One of the aspects that need to be addressed in FL is privacy, but focusing only on privacy can create an open door for attacks against robustness of FL process (resulting in creating a new attack surface) against security attacks. As a result, both the privacy and security aspects should be addressed simultaneously. In the literature, there are some solutions addressing these aspects, but they introduce considerable overhead or have some limitations such as the need of two non-colluding servers. To address this gap in FL, which is a promising tool for 6G, a new framework is introduced that allow the usage of any secure aggregation scheme and provides an approach to make the FL process more robust. Its performance on prevention of security and privacy attacks is tested by performing experiments and the computation and communication overhead of the proposed framework is analysed.

A brief description of the framework. Each client first divides its local model update into pieces, encrypts these pieces separately, and sends the results to the server. The server requests the clients to open some pieces. Each clients sends the requested pieces in cleartext with required parameters for encryption. Using the received cleartext pieces and parameters, the server performs encryption and validates the received cleartext pieces. After successful verification, the server performs some security attack detection/prevention mechanisms such as computing distance of each client's vector of cleartext pieces to other clients' vectors of cleartext pieces to detect if there is any abnormal behaviour as utilized in [HEX2-KPK+24]. Since the server cannot access the whole local model update, the server cannot infer information about the training data from the partially observed local model updates. The server can also perform security attack detection/prevention mechanisms by using the partially observed local model updates.

Experiment results on detection of security attacks. A simulation environment was established to test the performance of the proposed framework on detection of security attacks. Nine honest clients and one malicious client were simulated. According to the simulation, the malicious client added noise in form of gaussian distribution to its local model update. Each client had their training data set consisting of 20000 data records to be used to update the received global model from the server to generate local model updates. Only one linear layer weights were revealed to the server so that the server checks the distance between the weights vector of the linear layer. It is observed that the distance value for the malicious client can easily be distinguished as seen in Figure 4-9 where 5-th client is the malicious one.

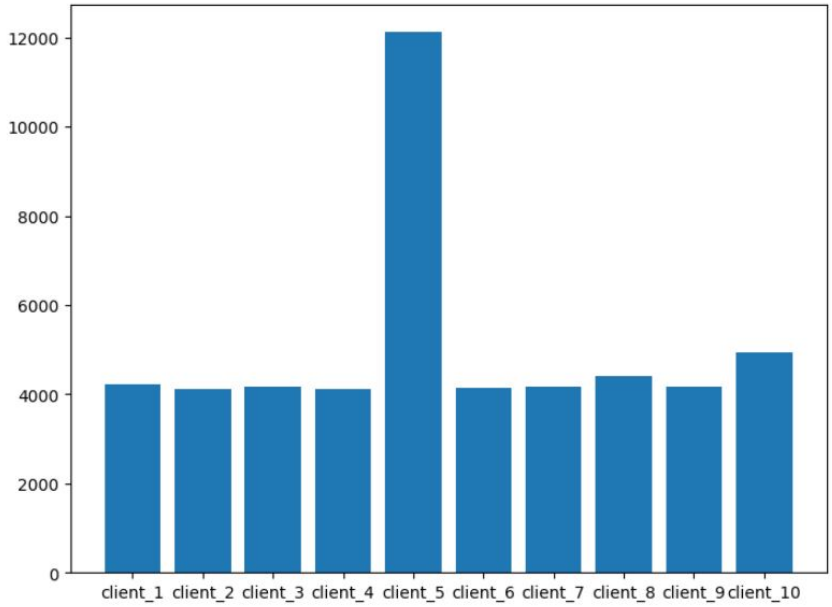

Figure 4-9: Weight vector distance [HEX2-KPK+24] between the local model updates from 10 clients.

Experiment results on privacy attacks. Experiments related to the application of Deep Leakage from Gradient attack [ZLH19] on FL with the proposed framework was performed. The same settings in [ZLH19] were used and PyTorch was chosen as the experiment platform. In the experiments, the LeNet convolutional neural network introduced in [LBB+98] was preferred. To simulate the proposed framework where the server can only access some pieces of the local model updates, some elements of the gradient matrix were removed (set to '0') and the success of the privacy attack was evaluated. Figure 4-10 demonstrates deep leakage from gradients which were shared throughout deep learning process. The attacker succeeded to create the ground truth image from the shared gradients starting from random init image. It is observed that from iteration 100 the loss value for prediction becomes zero and the attacker with high confident can predict the training data. Figure 4-11 presents the case where the proposed framework was used (i.e., some elements of the gradient matrix are set to zeros). As seen from the figure, the attacker was not able to construct the ground truth from random init image. As a result, it can be concluded that if some of the gradients are hidden then the attacker is not able to construct the training data and learn sensitive information.

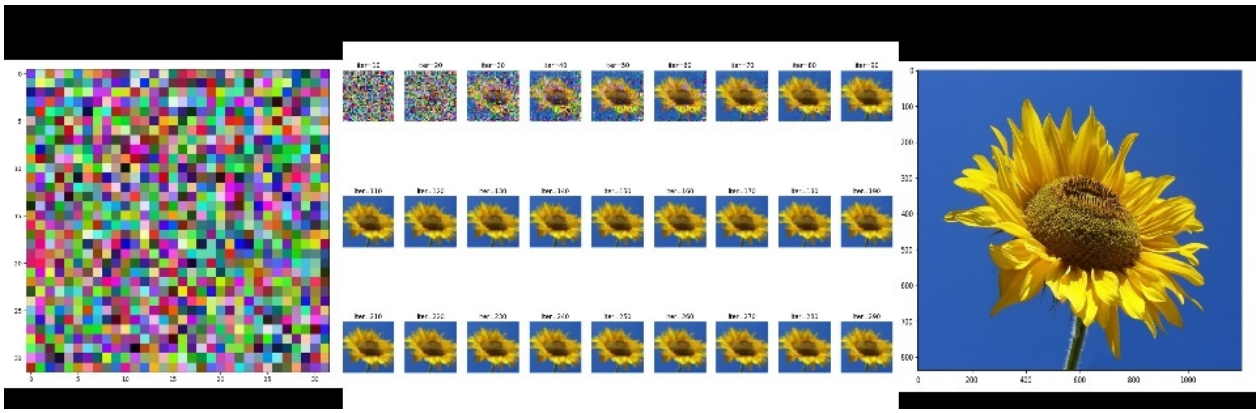

Figure 4-10: Deep leakage from gradients on an image when the server access whole individual local model updates. The images in the left- and right-hand sides are a random init and the ground truth images, respectively. The middle part shows the created images from the random init image after different number of iterations.

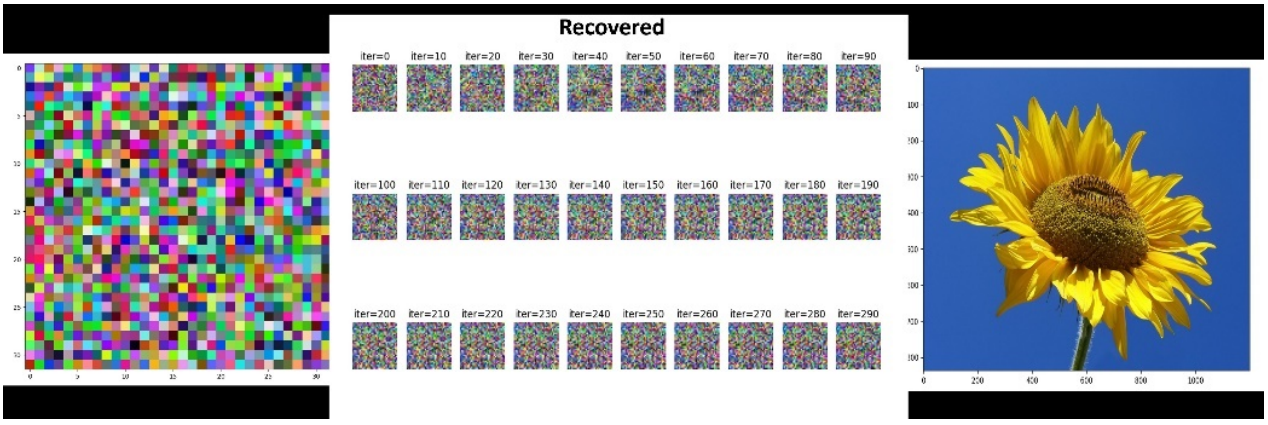

Figure 4-11: Deep leakage from gradients on an image when partially secure aggregation is applied on some of the gradients. The images in the left- and right-hand sides are a random init and the ground truth images, respectively. The middle part shows the created images from the random init image after different number of iterations.

Overhead analysis. The additional steps introduced by the proposed framework on top of a FL with secure aggregation are the following. The server sends the index set of the requested local model update pieces, and correspondingly the clients send the requested local model update pieces with required parameters for encryption. Since the overhead of these additional steps require sending only some parts of the local model updates in cleartext with required parameters, the communication overhead will be less than the cost of sending the local model updates in an encrypted format. Regarding the computation overhead, the server needs to execute encryption operation for the received cleartext local model update pieces. One considerable overhead in terms of communication can be the need of one additional communication between the clients and server. This requirement multiplies the required number of communication rounds by two, but this overhead can be welcomed when the advantages of the proposed framework are considered such as addressing both the privacy and security aspects simultaneously without requiring heavy cryptographic operations such as zero knowledge proofs and strong assumptions such as two non-colluding servers. Details of the framework and experiment results are available in [HEX2-KPK+24].

4.2.1.2 XAI-based security against adversarial attacks for intrusion detection systems

Vulnerabilities in the system allow attackers to perform DDoS attacks and increase their damage using sophisticated strategies such as adversarial attacks designed to harm Introspective Intrusion Detection System (IDS), which is an important concern. This exposure not only threatens IDS control, but also to sensitive data

and critical network processes. An attacker exploiting these weaknesses could launch multiple attacks that would overcome existing defences and ultimately break the defence mechanisms. Furthermore, as such incidents become more frequent, safety measures need to evolve to keep ahead of threats. The combination of DDoS and ML (Machine Learning) model targeted attacks creates a twofold threat that requires a robust and adaptive security response to mitigate the risk and protect the infrastructure. Solving the above problem requires a comprehensive framework that combines the strengths of ML against adversarial attacks, real-time applicability, handling complex attack vectors and ensuring scalability and robustness of explanations without human intervention.

A brief description of the XAI-based IDS framework. The proposed framework involves enhancing IDS to strengthen its defence against DDoS attacks that use adversarial techniques to evade detection. This strategy incorporates XAI to pinpoint specific characteristics manipulated by attackers, thereby exposing and accurately flagging altered network traffic aimed at tricking the IDS. By developing an XAI-based adversarial detection system for adversarial attacks, the system gains the ability to adapt and optimize its responses in real-time without the need for human intervention. Leveraging XAI techniques, the model can interpret the factors influencing its decisions, leading to an enhanced comprehension of its own functioning. This transparency allows for a deeper understanding of the behaviour of model, enabling it to autonomously fine-tune its approach for improved accuracy and performance.

Experiment results on adversarial attacks against IDS. During a DDoS attack, the use of spoofed IP addresses by the attacker reduces the number of matched flow entries, leading to an increase in the number of Packet-In messages. Therefore, a notable change in the rate of Packet-In messages is expected during a DDoS attack. This experiment takes advantage of this characteristic, utilizing the associated statistics of Packet-In messages as the primary features for detecting DDoS traffic samples. The flow of Packet-IN messages within a 10-millisecond interval is monitored, and their frequency-domain properties are analysed. The sequence of Packet-IN messages is essentially converted into frequency characteristics, which are then broken down into distinct frequency bins. These bins serve as unique features to differentiate between normal network traffic and potential DDoS attacks. The performance of the IDS is evaluated under two different conditions: a pure DDoS attack against monitored network to disrupt network traffic and a combined DDoS and adversarial attack. For this reason, an auto-encoder is utilized for the IDS. A DDoS attack is simulated by generating high volumes of traffic to overwhelm the network. The IDS is tested to see how well it detects and mitigates this attack. Later, adversarial examples are crafted to deceive the IDS to misclassify the malicious DDoS traffic as normal using techniques like Fast Gradient Sign Method (FGSM), projected gradient descent (PGD), and auto-PGD (APGD). While FGSM is a fast, single step method for generating adversarial examples by applying a small perturbation in the direction of the gradient of the loss function, PGD is a multi-step iterative extension of FGSM that applies successive perturbations, projecting the perturbed inputs back onto the valid data domain to enhance the effectiveness of the adversarial examples. APGD is an advanced version of PGD that automatically tunes step sizes and iteration counts to optimize the balance between attack strength and computational efficiency. Then the performance of the IDS is measured to identify the extent of degradation. The IDS performance, measured by metrics such as detection rate and false positive rate, will show a significant decline when adversarial attacks are added to the DDoS attack, highlighting the vulnerability of the IDS to adversarial manipulations. Figure 4-12 shows the accuracy of IDS under different adversarial attack methods as a function of epsilon, which represents the perturbation strength used in the attacks. X-axis represents the magnitude of the perturbation applied to the input data during the adversarial attacks. A higher epsilon value indicates stronger perturbations. Y-axis represents the accuracy of the IDS in correctly classifying inputs as either normal or malicious. No attack shows the baseline accuracy of the IDS when there are no adversarial attacks. As epsilon increases, the accuracy drops, showing the performance of IDS degrades with stronger adversarial perturbations. FGSM accuracy decreases more sharply compared to PGD as epsilon increases, indicating that FGSM is more effective at degrading the IDS performance with increasing perturbation strength. APGD is like PGD, the accuracy drops significantly with increasing epsilon, but the curve shows a less consistent degradation. Overall, FGSM shows more severe impacts compared to PGD and APGD as the perturbation strength increases.

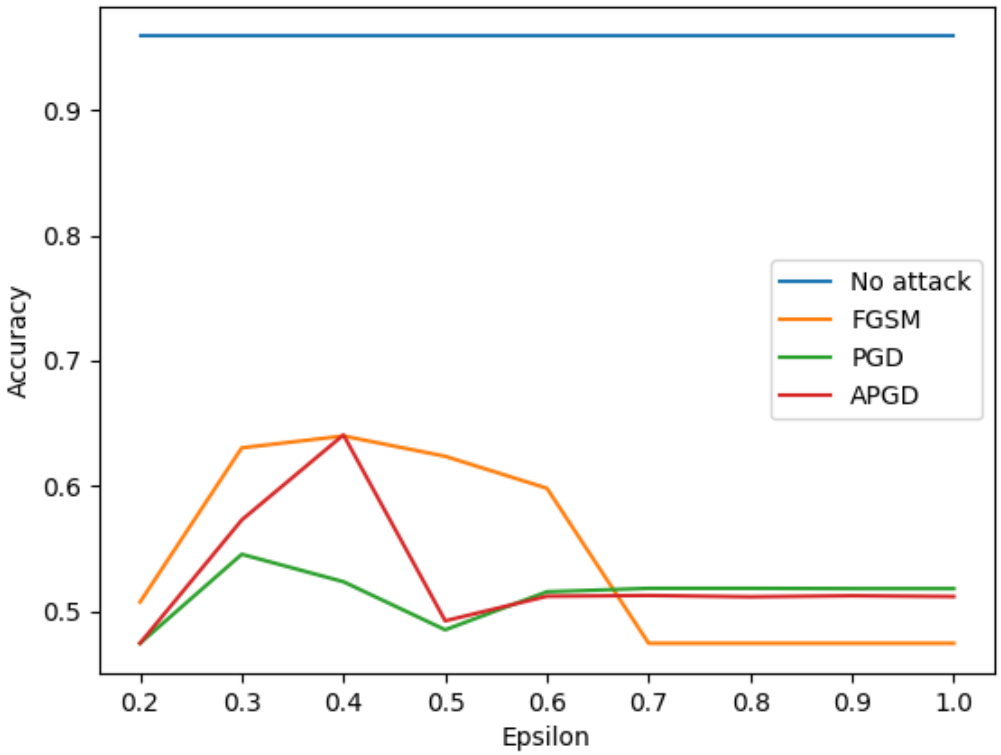

Figure 4-12: The accuracy comparison of different attacks against IDS.

Experiment results on detection of adversarial attacks against IDS. An adversarial detection method is proposed that leverages XAI to differentiate between original data coming out of feature engineering module and the data modified by adversarial attacks. SHAP is used to extract and visualize the features of both the original and attacked data. This process helps in understanding how the adversarial examples differ from the original traffic. The detection method is built to recognize the patterns indicative of adversarial modifications. This way, the system will learn to distinguish between normal, DDoS, and adversarial traffic based on the extracted features. Figure 4-13 shows the list of important features, from most significant to the least significant one (top to bottom). Each shape shows the contribution of each class in the certain feature in the prediction. Features 2, 3 and 5 are the one with the most predictive power according to both models. However, after an adversarial attack the ranking of the features are changed. In addition to rankings, while some features are not important originally, after attack, the model is manipulated, and Features 57, 58 and 59 become important. This clear difference between SHAP features proves that there is an additional adversarial attack and by checking this difference the system can send an alert to IDS.

Experiment results on mitigation of adversarial attacks against IDS. As a mitigation approach, simple yet effective method is proposed by setting thresholds for the number of different features identified by the XAI methods in the feature engineering and the intrusion detection module. Features from incoming traffic is continuously extracted using the XAI-enhanced IDS in a run-time. A threshold for acceptable variations in the key features is identified. This threshold is based on the observed distributions of features under normal and DDoS conditions. Finally, a monitoring mechanism is implemented that triggers alerts or blocks traffic when the number of features exceeding the established threshold surpasses a predefined limit. This mechanism serves as an additional layer of defence to catch adversarial attacks that surpass the primary detection system. Figure 4-14 compares the accuracy of the IDS under various conditions: no attack, different adversarial attacks (FGSM, PGD, and APGD), and their proposed corresponding mitigated versions. All adversarial attacks significantly degrade IDS accuracy, with auto-PGD having the most severe impact. The threshold-based mitigation method effectively reduces the impact of adversarial attacks by flagging anomalous traffic patterns that deviate significantly from the norm, thereby preventing significant degradation in IDS performance, but do not fully restore the IDS performance to the baseline level. This comparison highlights the need for robust mitigation strategies to enhance the resilience of IDS against sophisticated adversarial attacks.

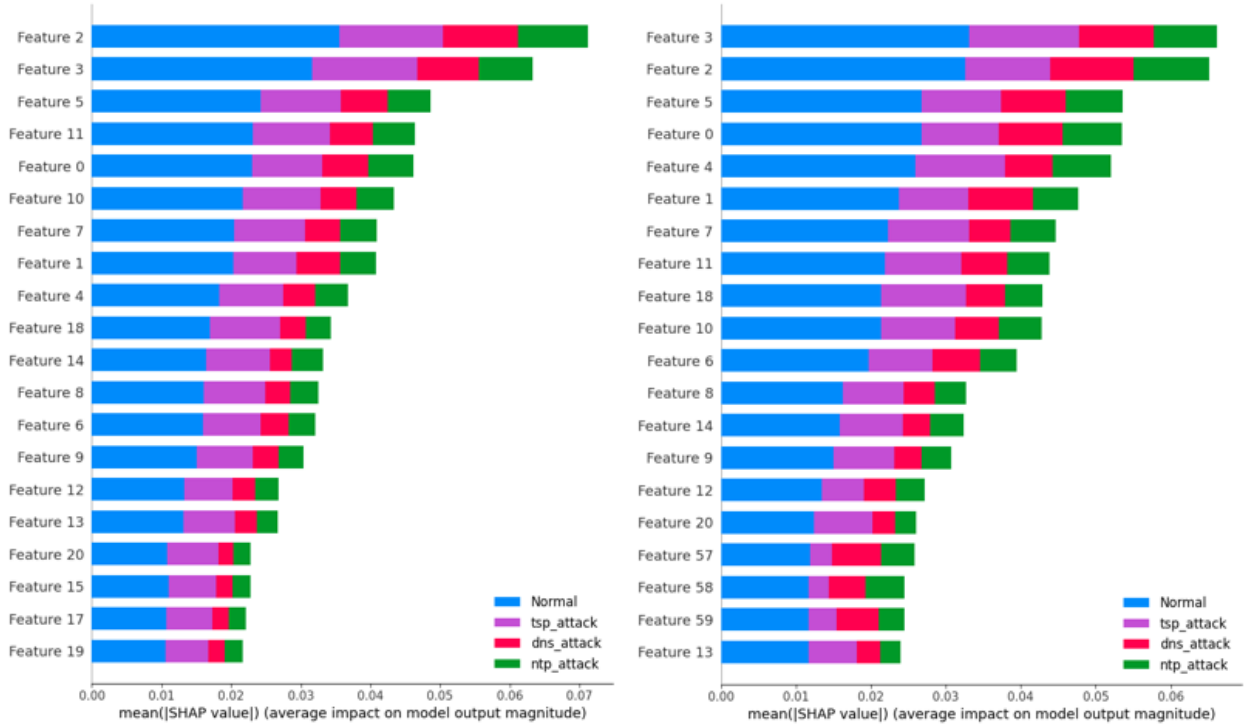

Figure 4-13: Relative performance of the twenty most important features before and after attack.

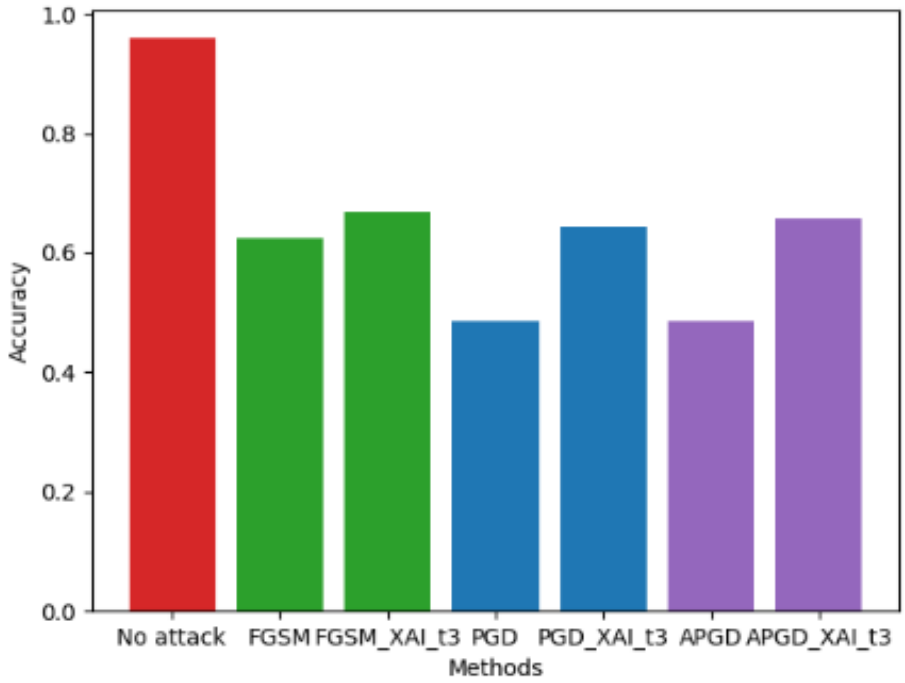

Figure 4-14: Method comparison with XAI-based mitigations.

Overhead analysis. The incorporation of XAI into IDS represents a significant step forward in strengthening cybersecurity defences against DDoS attacks employing adversarial techniques. By leveraging XAI to enhance the transparency and adaptability of the detection system, the proposed framework enables more accurate and efficient identification of anomalous network behaviour indicative of adversarial activity.

4.2.2 Quantum-Resistant Cryptography

The transition to quantum-resistant cryptography is currently led by exploring the mechanisms for applying Post-Quantum Cryptography (PQC) algorithms, and their impact in current practices for protocol security, especially in terms of availability, performance and the use for establishing trust among the communicating peers.

This analysis is starting by means of applying an open-source PQC evaluation platform, in cooperation with the Horizon Europe cluster on PQC [SPQR24]. The evaluation platform selected is Qujata [Quj24], being developed to evaluate the performance of quantum-safe cryptographic protocols and compare them.

Qujata works by creating experiments whose parameters include the desired algorithm(s) to evaluate, message sizes and the number of iterations. Algorithms include post-quantum ones (Bikel, Frodo, Hqc and Kyber in different versions), classical ones (Prime256v1 and Secp384r1) and hybrid versions, in order to also evaluate transition strategies. The Qujata architecture is designed to accommodate new algorithms and integrate with other input/output elements by using virtualization, as experiments themselves are run using either Docker Compose or Kubernetes with Helm charts.

Qujata claims it is able to analyse the performance of algorithms by "client and server vital signs, like memory and CPU usage, connection time, download speed, once connection is established, etc", though the current distribution of the library offers the possibility of measuring just several indicators. More particularly, it offers:

- Average CPU: one of the inputs that the API accepts is the number of iterations to perform. For each instance of iterations given to Qujata, it will compute the average CPU usage of the experiment.
- Average memory: similarly, it will compute the average memory usage of the experiment.
- Bytes throughput: in bytes/second.
- Request throughput: in messages/second.

The number of indicators needs to be expanded to encompass other metrics in the future, and work is ongoing within the project on this expansion. The data can be visualized both using a table format or via several line and bar graphs that, for each experiment, and allowing the user to compare all experiments simultaneously. Moreover, Grafana can be used to visualize the data and analyse the performance of the task over time.

The current Qujata API is programmed in Python, relying on a Docker image to perform the experiments, which currently is the latest version of openquantumsafe/nginx, from the Open Quantum Safe (OQS) project [OQS24]. Leveraging the Qujata fully virtualized approach, work is ongoing to explore additional algorithms, analyse hybridization approaches, including strategies for incorporating Quantum Key Distribution (QKD) mechanisms. The intended Qujata evolution also considers additional metrics, beyond the current measures on algorithm runtime (processor cycles and memory consumption) better suited to evaluate the impact on commonly used secure protocols, such as the induced latency and the deviations in execution time that can induce jitter.

4.2.3 E2E Resilience Evaluation

This section provides examples of two kinds of simulations suitable to be executed to compute network availability KPIs, according with different operating circumstances.

4.2.3.1 VNF availability

In a first step, we consider a system with one virtual network function (VNF) composed of two virtual microservices, and we assume these microservices can be deployed by the Kubernetes platform on a Data Center with three available servers. Each microservice contains exactly one container, and only one container is deployed on a Kubernetes pod. All pods are deployed on nodes that are physical servers.

Two types of failures, the physical failure on servers (nodes) and the software failure on microservice containers (pods), are chosen as the main risks to the VNF sub-model. The two kinds of failures occur randomly. The failure times of these two failures follow exponential distributions.

The pod repair process is automatic and is managed by Kubernetes. Kubernetes throws a liveliness detection probe to check the running status of a pod at every health check interval.

Regarding a node failure, the self-healing process is slightly different. A node is a physical server. A node cannot be terminated. The technical team will manually repair the failed server, and afterward, the repaired server will become a free server available for use in the Data Center. A node failure also leads to failure of the pods deployed on the node. Those pods will be replaced by new ones on other available nodes.

Let's study the effect of self-healing detection frequency on system availability. The two microservices are assumed to be managed by the same Kubernetes Master (pod liveliness detection and node liveliness detection are synchronized and done at the same time). The numerical solution from analytical results is compared with the simulation results from a Python program based on SimPy platform. All simulation parameters are given in Table 4-.

Table 4-4: VNF parameters defined for the E2E Resilience Evaluation VNF simulation [LDB+22].

Parameter	Value
Pod failure	MTTF (Mean Time to Failure) = 1258 hours
Pod termination time	30 seconds
Node failure	MTTF = 8760 hours
Average time for pod instantiation	5 seconds
Average time for node creation	1 second
Node capacity	3 pods per node
Data Center capacity	3 servers (nodes)
Self-healing probe period	0 (immediate), 2, 5 and 10 seconds

We simulate the VNF behaviour over 50 years (sufficient large arbitrary choice for failures to occur). The average value of microservice uptime and VNF uptime over 20000 simulations is taken as the final result. After 20000 simulation iterations, the results from the SimPy-based Python program converge well and took only few minutes on a computer equipped with Windows 10, 2.10 GHz CPU, and 8 GB memory.
The result is shown in Table 4-. We compare the overall VNF availability for health check interval varying from 0 to 10 seconds. The longer the probe period, the lower the overall availability. The availability drops from 5 nines to 4 nines by changing immediate detection to 10 seconds.

Table 4-5: Analytical and simulation results comparison.

Detection time (probe period)	0 (immediate)	2 seconds	5 seconds	10 seconds
Analytical	99.9992911%	99.9991494%	99.9989367%	99.9985823%
Simulation	99.9992911%	99.9991491%	99.9989364%	99.9985821%

Simulation results are validated for modelling a VNF for 5G or 6G network.

4.2.3.2 Vertical use case

In this second step, let's consider the Tele-action use case originally proposed by the 5G EVE project [5GE18-D11]. It refers to remote decoupling protections for the distributed generation power stations in the electric grid. The use case is presented in Figure 4-15. The integration of distributed generators into distribution networks affects the requirements and the performance of conventional protection schemes.

It may cause problems such as unwanted islanding of feeders or unwanted disconnection of distributed energy resources during high voltage faults or wide area disturbance, which may cause damages. This problem can be solved thanks to Tele-action.

In Figure 4-15, the considered network includes four distributed energy generation units and one customer load. They connect to the high-voltage network via two feeders.

The whole process of the scenario during a network fault is given as follows:

- At the beginning, the distribution network is in the normal operation mode.
- A fault situation occurs, leading feeder 1 to open the protection.

- The unwanted islanding operation will be avoided by remote decoupling through signalling to the distributed generation stations connected to the segment that feeder 1 controls.

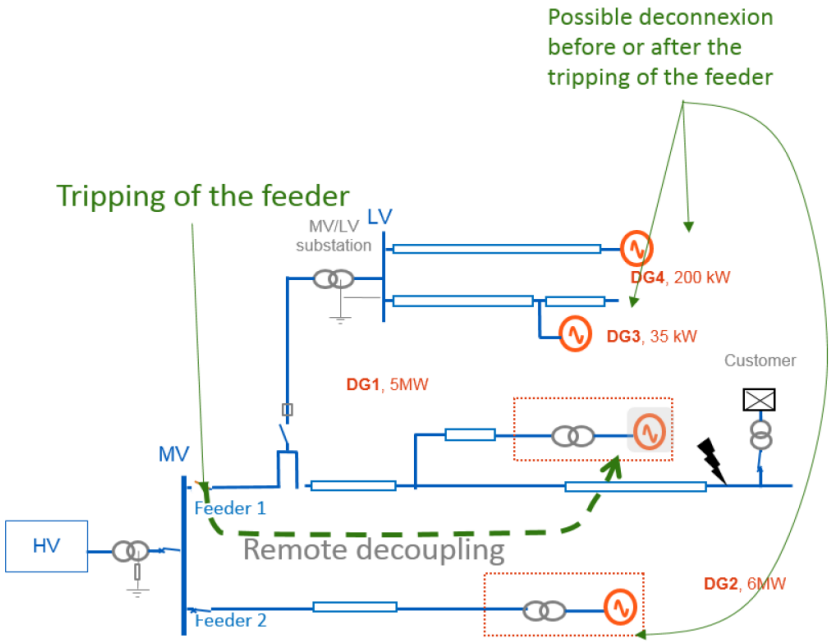

Figure 4-15: Tele-action use case presentation.

Some KPIs have been targeted in the 5G Tele-action use case:

- The one-way latency should be less than 50 ms from device to device. (dotted-line "remote decoupling" arrow extremities)
- The network availability should be above five nines, i.e., 99.999%.

Let's zoom on the availability KPI. For the simulation tool, we will consider the essential functions and build the corresponding Service Function Chain (SFC). The User Plane part is critical for such a service. The simulation will be based on an SFC with O-RAN virtualized functions in the access (DU and CU) and UPF function for the core network. Routing diversity in the transport network is assumed in the simulation. Probe period is assumed to be immediate.

The simulations results showed that with a single antenna attachment is not enough to reach vertical expectation. The availability is estimated at 99.9731 % by the simulation tool. If End-to-end path diversity can be ensured, the availability becomes 99.9999 % and requirements seem fulfilled. In the simulation tool, radio part is not included but 5 nines target should still be reached.

4.2.3.3 Next step regarding risk mitigation

This simulation tool can help to mitigate risks assessed in D1.3 [HEX224-D13] for some use cases families. For example, the implementation of network-assisted mobility solutions may inadvertently contribute to unanticipated traffic congestion and thus counteracting environmental goals. The proposed mitigation is to continuously monitor and adjust traffic management strategies to minimize congestion and optimize environmental impact. With the resilience simulation tool, several traffic scenarios can be run to check if the dimensioning and auto-scaling mechanisms are enough to face the traffic increase or if additional resources are required.

4.3 Enablers at Development TRLs

These enablers are associated with technologies in development the project is considering to become part of the 6G security toolbox, to address the *6G delta*. This section provides results of initial experiments evaluating the conditions and cases for the application of these technologies, as well as the information model to be applied in the specific control interfaces, and considerations on their use.

4.3.1 Confidential Network Deployment

Two different SPR controls are being considered here. Apart from discussing them in the enclosed subsections, the possibility of a joint OpenAPI interface to both SPR controls should be considered here.

4.3.1.1 Confidential Computing

[HEX224-D23] included details about the Confidential Computing experimental setup, which confirmed the principal applicability of Confidential Computing for the deployment of CNFs. In this deliverable, an update about performance, operations, and standardization aspects will be provided.

Performance

The analysis is specific to the usage of Intel SGX as trusted execution environment (TEE). Similar results are likely also for other TEEs, but this still needs to be confirmed with additional experiments.

Execution speed of code within the CPU is independent from whether it is executed inside or outside a TEE. However, performance overhead originates from transition to and from an enclave and from memory access [Int18].

The impact of transition to and from an enclave is like what happens during normal context switches (i.e., when the CPU for the sake of multi-tasking is switching from one process to another). Enclave transitions come on-top of these context switches. In case of a confidential CNF these transitions typically occur, if IP packets are sent to and from an IP stack terminated within the CNF's enclave. To quantify this overhead simple http client and server test applications have been used, which are either running inside or outside of an enclave, respectively. In case of deployment within enclaves the performance of a http GET request/response (with empty response body) increases from 0.45 ms to 0.78 ms (i.e. by a factor of 1.7).

Modern CPUs are using a cache for memory access. That is, instead of reading from and writing to main memory, the CPU uses a cache. If data cannot be read from cache, a cache miss occurs, and the data is loaded from main memory to the cache. In case of confidential computing the cache is not directly filled from main memory, but from a special enclave cache (known as Enclave Page Cache, EPC). If this enclave cache is not sufficiently large paging to the main memory occurs. Thus, in case of confidential computing two performance penalties related to memory access exist. The first one is related to cache misses and the second one related to enclave paging.

A small test program has been used to quantify both overheads. The test program allocates a matrix with variable size, fills the matrix with data, and subsequently calculates the sum of all matrix elements. The test program is either executed as a normal application or within an enclave. If the matrix fits into the cache no difference in performance occurs. If with increasing matrix size cache misses occur, the performance of the code inside the TEE goes down by a factor of 8, whereas the code executed outside of the TEE only shows a small performance degradation of about 20%. If the matrix size is increased further and enclave paging starts to occur (because the matrix does not fit into the EPC), the performance of the code in the TEE is going down by another factor greater than 10, whereas the performance of the code outside the TEE remains rather stable. Thus, if enclave paging occurs, execution within a TEE slows down execution by a factor larger than 100. Especially in distributed setups such a severe performance impact can easily lead to message timeouts, retransmission, and finally a system breakdown. Thus, CNFs need to be designed in such way, that enclave paging does not occur.

To better understand how these performance degradations related to confidential computing potentially influence the deployment of CNFs, the time for initial registration of a UE has been measured for deployment of an open source 5G core either as a confidential deployment or a standard deployment. In both cases the UE and the gNB have been simulated. For the confidential deployment all 5G NFs involved in the call flow (e.g., access and mobility management function, AMF, authentication server function, AUSF, unified data management, UDM, unified data repository, UDR, and policy control function, PCF) have been deployed within a TEE, each.

As a main result the time of initial registration of the UE is increasing from 80 ms for normal deployment to 270 ms for confidential computing, i.e., by a factor of 3.4. Only to a smaller extend this performance impact can be attributed to the large number of messages in the call flow (which each include transitions to and from

enclaves). The larger fraction of the overhead is related to cache misses. Furthermore, the performance was obviously not impacted by enclave paging.

Since the performance penalty related to confidential computing, is largely impacted by cache misses, results might vary, if different hardware with a larger cache is used. The cache size of modern CPUs is typically increasing, but the same is true for the number of cores per CPU. That is, the amount of cache per core is increasing to a much lesser extent (if at all). In general, code used in CNFs might be prone to cache misses (and thus to performance degradation related to confidential computing). Thus, the selection of the right hardware but also the evolution of the hardware towards optimized memory access is an important aspect related to the efficient usage of confidential computing for CNFs. In addition, more analysis is needed, if a different design of the CNFs or even of the network architecture could facilitate data locality and thus also the performance of confidential deployments of CNFs.

Operations

From an operations point of view the most important task is to configure the key management service (KMS) with mappings between the attested CNF and the secrets to be provisioned. The secrets could be either final secrets (like private keys and X.509 certificates to be used for transport layer security (TLS) communication) or bootstrapping secrets (like tokens), which can be used to obtain the final secrets from another provisioning server. This is a design question, which might be addressed by standardization bodies later. Note, that the attested CNF is not identified by the container image, but by the enclave identifier, which is included in the evidence presented by the CNF to the KMS. This identifier could either be the hash of enclave or the identifier of the enclave signer (typically the CNF vendor) and a product id. In any case supply chain management needs to be extended to securely deliver these identifiers from the CNF vendor to the operator. Details are for further study.

Another important aspect is the possibility to use the same operational procedures for different hardware potentially offering different TEE approaches and for traditional deployments without using TEEs. That is, the way how the KMS is configured with mappings between CNFs and secrets and the protocol between KMS and CNF should be independent from whether a TEE and which TEE is applied, but only the evidence presented by the CNF to the KMS varies depending on the used TEE approach. Ideally also the same CNF container image could be used independent from the used TEE approach, but it is for further study, if this is indeed feasible from a technical point of view.

Standardization

Subsequently different standardization strategies for the various interfaces are discussed. The naming of the interfaces and their role in the architecture is explained in Figure 4-16. Due to some technical uncertainties no final conclusions can be and should be drawn yet.

Interface between KMS and CNF (K-1):

The K-1 interface is used by the CNF to present evidence about its enclave to the KMS and in turn (after the KMS has verified the evidence) to provision the assigned secrets to the CNF. Different approaches exist for its standardization.

In one approach the interface between KMS and CNF could be specified. In this case, the vendor of the KMS as well as the vendor of the CNF can implement their products against this interface. In this case the internal interface between the telco part and the *init* function (i.e., I-1) is not relevant and each vendor of the CNF could decide whether it pursues the concept of an *init* function.

Alternatively, a KMS might be based on a proprietary interface. In this case the KMS could provide the code the *init* function to the CNF vendor (either as source code or as a binary). In this way the CNF vendor can create the CNF's enclave, which includes own code and the *init* function provided by the KMS vendor. The drawback of this approach is that the CNF vendor needs to maintain different CNF images depending on the used KMS. Furthermore, it should be kept in mind that generation of the CNF's enclave is typically using a certain TEE framework and the *init* function provided by the KMS vendor might not be supported by or compatible with the TEE framework preferred by the CNF vendor.

Interfaces specific to remote attestation (A-1, A-2)

The two key capabilities enabling remote attestation are the ability of the CNF to collect evidence about itself using A-1 interface and the ability of the KMS to verify with the help of the attestation server the evidence (e.g. that it indeed refers to a valid TEE) using A-2 interface. To some extend these capabilities might be standardized by the Internet engineering task force (IETF). However, since the evidence and the way how it is verified strongly depends on the used CPU hardware it remains to be seen, how feasible this approach will be. If the interface used to collect the evidence will not be standardized in a CPU vendor agnostic way, the CNF needs to handle these hardware dependencies, which might be a major obstacle. Likewise, if the interface between the KMS and the attestation server depends on the used TEE, the KMS needs to be able to handle this to avoid a limitation of the choice of available TEEs.

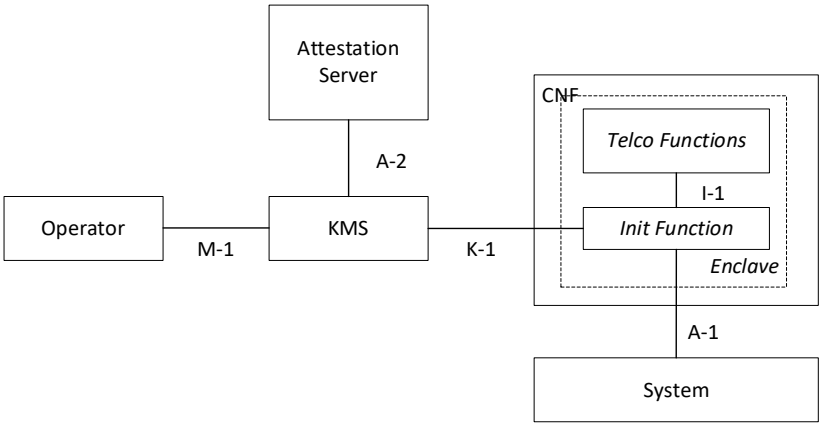

Figure 4-16 Schematic outline of the interfaces used for Confidential Computing.

KMS Management Interface (K-1)

The M-1 interface is used by the operator to configure the binding between secrets to be configured to the CNF (identified by hash or signature of its enclave). Most likely, standardization of this interface only makes sense, if the K-1 interface is standardized. In this case the same standardization organization might take care about the standardization of both interfaces, i.e., M-1 and K-1.

4.3.1.2 Topology Attestation

The impact on network performance of topology attestation has been evaluated following the experimental setup described in section 6.2.2 of D2.3 [HEX224-D23] and included in Figure 4-17 for reference, based on a P4 implementation of the Ordered Proof of Transit (OPoT) mechanisms. Several experiments were conducted to measure the system performance. We provide here a summary of the more detailed results, reported in [VEP+24].

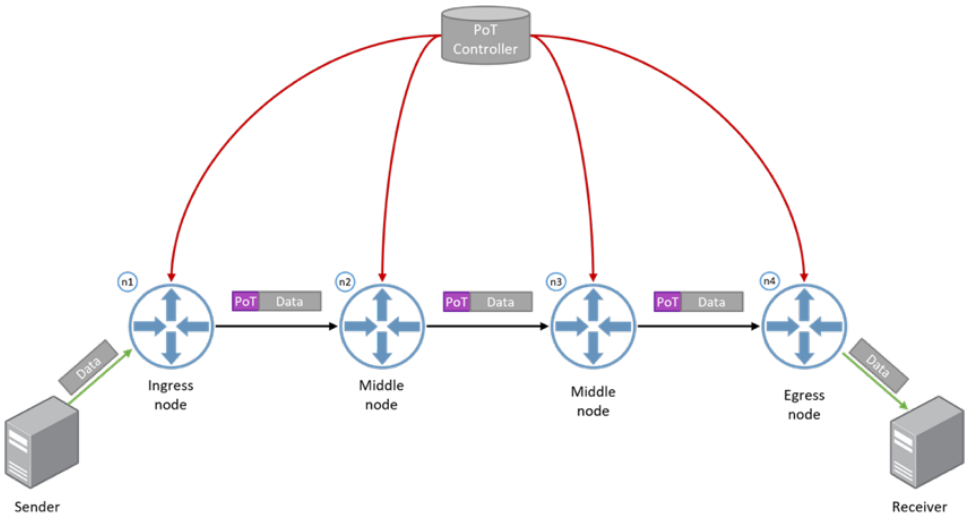

Figure 4-17: Topology for OPoT evaluation, as depicted in [HEX224-D23].

In all cases, a number of packets is transmitted between two hosts at the endpoints of the network., and a dedicated database is used to store metrics related to packet forwarding.

The tests started with a verification of proper forwarding through the four OPoT-enabled nodes. After this initial test, a similar test was conducted in a malicious scenario, where packets did not reach their intended destination, as each one of them was dropped by the egress node due to the presence of a middle forwarder not correctly performing the OPoT calculations.

The performance of the scenario is compared to an equivalent scenario with P4 nodes configured for basic packet forwarding without OPoT functionality. The paper also analyses the impact of frequency of metric generation and transmission to the collector. Finally, the impact on performance of the number of nodes in the OPoT-enabled chain was also evaluated.

To determine the direct impact on performance caused by OPoT compared to normal traffic behaviour (Figure 4-18), latency and throughput are measured and compared with the same measurements made in a scenario with four nodes, where each node is configured with a P4 program to operate based on basic forwarding behaviour (Figure 4-17).

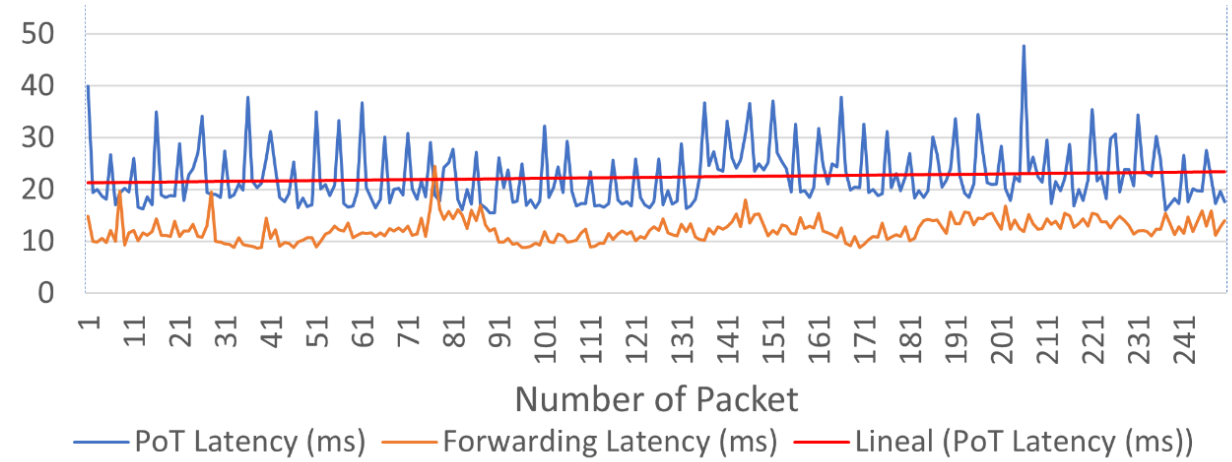

Figure 4-18: Latency Comparison for OpoT and direct packet forwarding.

As shown in Figure 4-18, the observed latency penalty, associated to the computational overhead incurred by OPoT procedures, indicates that OPoT cannot be universally applied to every packet but rather by sampling or selecting those flows or applications where the use of OPoT is critical.

Furthermore, Figure 4-19 displays the impact of the number of nodes in the OPoT-enabled chain, with results obtained by ranging from four to ten nodes.

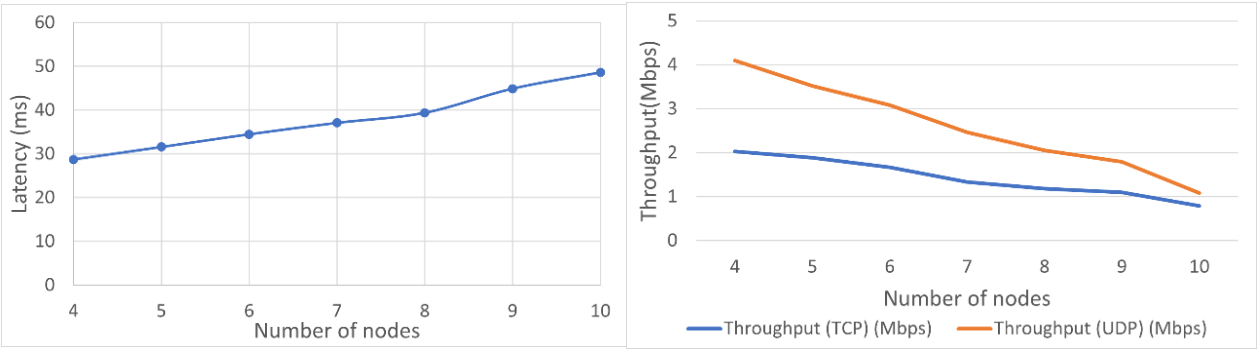

Figure 4-19: Latency and throughput performance of OPoT chains based on number of scenario nodes.

It is noticeable that all graphs follow a linear trend, whatever the transport protocol in use. This linear behaviour facilitates the assessment of the impact of the length of possible OPoT-enabled chains on packet forwarding performance.

As it is clear from the proposed diagram, the use of OPoT mechanisms implies the attachment of the ingress and egress nodes of the OPoT-enabled chains to the endpoints of a communication link. This pattern implies the identification of the addresses of these nodes and the provision of the appropriate path, typically by means of a SDN control interface. The control interface for the OPoT-enabled chain should contain specific operations for commissioning and decommissioning the chain. The commissioning operation must include the appropriate information elements to configure the OPoT mechanisms, namely:

- Egress and ingress addresses
- Intermediate node identifiers (addresses, references in a service forward chain…)
- Service path identifier for the forward and backward paths
- Sampling rate (zero implies applying OPoT to all packes)
- Level of metrics to be collected (none, at egress/ingress nodes, at all nodes in the path)
- Refresh rate for crypto material (Shamir coefficients and masks)

4.3.2 Distributed Ledgers

The increasingly interconnected nature of modern network architectures demands advanced security and trust mechanisms, particularly in environments spanning multiple technology and stakeholder domains. Traditional approaches to network management systems, while effective in static settings, often lack the flexibility, security, and transparency required for dynamic and decentralized operations typical of multi-domain configurations. Distributed Ledger Technologies (DLTs) offer a robust alternative by enabling decentralized management and immutable recording of network configurations, inherently increasing security against tampering and unauthorized changes. This work introduces a DLT approach utilizing a private, permissioned ledger where topology changes are recorded as transactions.

The setup enforces data integrity and restricts access to network topology information, ensuring only authorized stakeholders can make changes. Consequently, it enhances security, maintains data consistency, and builds trust among network components by keeping a verifiable record of all changes. Additionally, this work examines the performance and operational efficiency of integrating DLT within network systems, using the CTTC ADRENALINE testbed [MNC+17]— an advanced infrastructure for Beyond 5G and future 6G services— as the platform for analysis. Experimental results reveal the system's performance metrics, illustrating the practical viability of implementing DLT in network operations management.

The proposed integration of the DLT testbed aims to enhance the security, transparency, and trustworthiness of network topology management across multiple stakeholder domains. Our proposed model, depicted in Figure 4-20, integrates DLT with nodes distributed among the controllers: the IP SDN Controller, the Optical SDN Controller, and the E2E SDN Orchestrator.

The ADRENALINE testbed is an open and disaggregated SDN/NFV-enabled packet/optical transport network and edge/cloud computing infrastructure designed for Beyond 5G, future 6G, and IoT/V2X services. It encompasses several network segments, including the edge/access and metro networks, integrating advanced network technologies to provide a flexible and scalable infrastructure. The testbed features a partially disaggregated optical network functioning as the metro network, comprising a photonic mesh network with Reconfigurable Optical Add-Drop Multiplexer (ROADMs) and bidirectional Dense Wavelength-Division Multiplexing (DWDM) amplified optical links and a Spatial Division Multiplexing (SDM) domain with Spatial Cross Connect (SXCs) devices connected by multi-core fiber. It includes packet optical nodes with 400G data rate transponders for transporting traffic flows between the access and metro network segments. IP Cell Site Gateways (CSGs) represent heterogeneous access network technologies with Edge DC capacities and connections using 100G QSFP28 interfaces.

The ADRENALINE testbed is managed by dedicated TeraFlowSDN controllers for each technological domain (e.g., IP, optical) and coordinated hierarchically through an E2E TeraFlowSDN Orchestrator, supporting seamless end-to-end connectivity and efficient resource orchestration across diverse network segments.

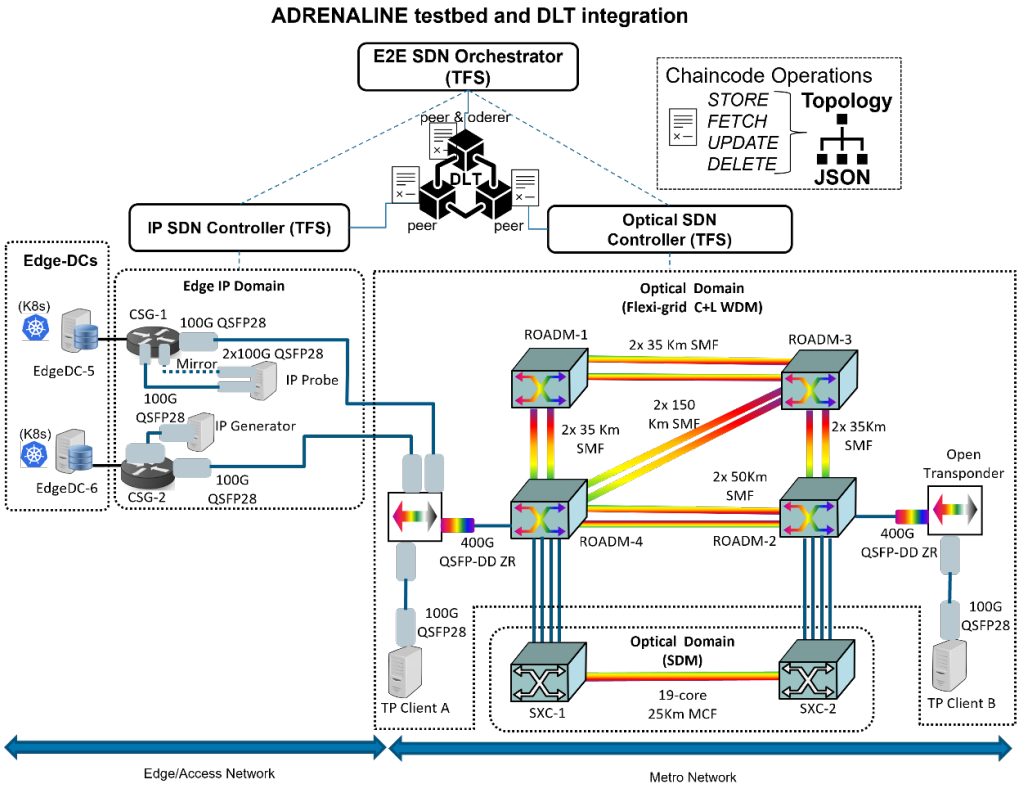

Figure 4-20: Proposed model integrating DLT with the ADRENALINE Testbed.

In the proposed methodology, DLT guarantees the integrity and security of topology management operations within the ADRENALINE network. This approach leverages smart contracts to implement the chaincode logic for operations such as STORE, UPDATE, FETCH, and DELETE topology information. Utilizing a permissioned blockchain framework such as Hyperledger Fabric ensures that only entities with the appropriate access rights can interact with the chaincode and perform these operations. Hyperledger Fabric's Membership Service Provider (MSP) manages identities and access control to authenticate and authorize participants.

The decentralized nature of the blockchain distributes the storage and operations regarding network topology information across multiple peer nodes, eliminating single points of failure and ensuring resilience. Channels provide isolated environments for transactions, ensuring privacy and data integrity by allowing only authorized participants to access specific ledgers. The world state database maintains the current value of the ledger's data, ensuring the most recent network topology information is readily available for all authorized participants.

The DLT integration for topology management is depicted in the sequence diagram shown in Figure 4-21. Initially, the IP SDN Controller and Optical SDN Controller store their respective topologies by sending STORE requests as transaction proposals to their corresponding peer nodes which endorse the transactions and forward validated transactions to the Orderer. The Orderer delivers a block containing the ordered transactions back to the peer nodes, which record the topologies in the DLT Ledger and confirm the operation to the controllers. Subsequently, the E2E SDN Orchestrator sends FETCH operations to retrieve the IP and optical topologies by querying the peer nodes, which query the DLT Ledger and return the topology information. The E2E SDN Orchestrator then computes the end-to-end topology and STOREs this aggregated information in the ledger, sending a transaction to the blockchain, recording the data, and confirming the operation.

The experimental setup was configured as illustrated in Figure 4-22. A Hyperledger Fabric v2.5 deployment was prepared on three Ubuntu 22.04 machines housing the E2E SDN Orchestrator (E2E TFS), Optical SDN Controller (Optical TFS), and IP SDN Controller (IP TFS) with full network visibility between each other. The deployment was configured with one orderer node and three peer nodes. The orderer and peer1 nodes were positioned in the E2E TFS machine, while peer2 and peer3 were positioned in Optical TFS and IP TFS respectively. The Hyperledger Fabric components were deployed as Docker containers with networking managed directly by a configured Docker network. Host naming was configured on each machine for domain name resolution.

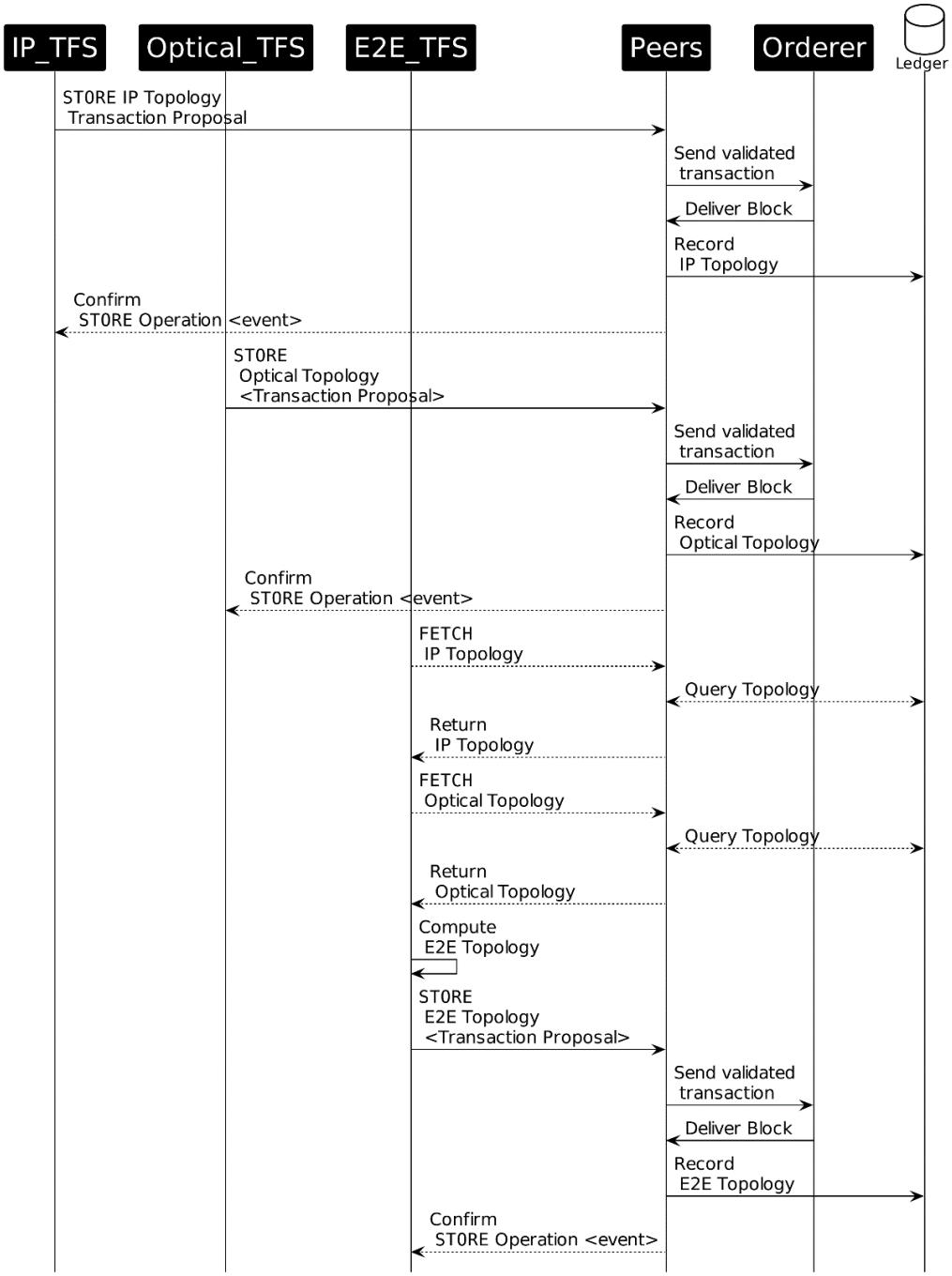

Figure 4-21: Topology management DLT operations.

The chaincode logic for the proposed experimental scenario was coded in Golang with functions for STORE, UPDATE, FETCH, and DELETE operations on network topology data. A single channel and organization were configured for the deployment, with all nodes belonging to the same organization. Consequently, the chaincode was installed and activated on all the peer nodes so they could endorse transactions.

Four topologies that are part of the ADRENALINE testbed were utilized for the experiments. The topologies are JSON files with key pairs for devices and links alongside a topology_id. As the transactions per second in a DLT are affected by the size of the managed data, the size of each topology object is given as a reference. The sizes of the JSON files in kilobytes are as follows: topo_dc with 5 kB, topo_ip with 16 kB, topo_e2e with 18 kB, and topo_optical with 10 kB. The methodology for the results considered a test scenario in which a Node.js application writes and fetches topology information from the blockchain. The codification of the Node.js application follows the guidelines of release-2.5 of Hyperledger Fabric, where interaction with the chaincode is achieved by the gRPC class to the Fabric Gateway API that exists on the peer nodes.

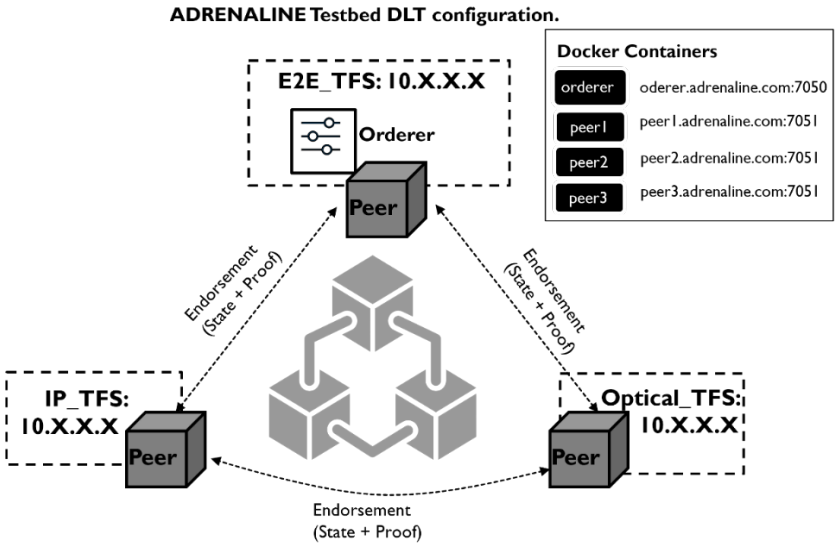

Figure 4-22: DLT experiments configuration.

The performance of the chaincode operations was evaluated by running the Node.js application and executing each operation 1000 times for a given topology. Figure 4-23 showcases the performance results as a histogram condensing the chaincode operations for every topology. The operations with the higher execution times are those that write/change the state of the ledger, while the operations that query the blockchain are executed almost instantly. Despite the variation in the size of each topology, there was not enough variation in execution time for each operation.

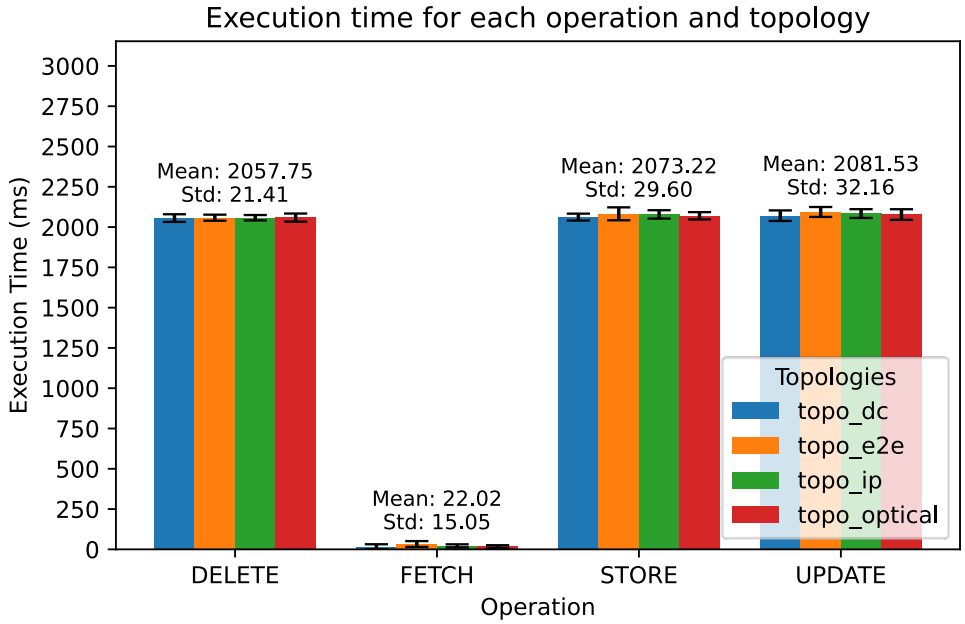

Figure 4-23: Execution time for each operation for all topologies.

Figure 4-24 presents the CDF of the execution times for each chaincode operation within the topology topo_e2e. This topology was selected as the size of the JSON file was the largest. The CDF plot clearly shows that the operations exhibit distinct clusters of execution times along the x-axis. Operations that query the blockchain, such as FETCH, are represented in green and demonstrate very low execution times, clustering near the origin of the x-axis. This indicates that these operations are consistently completed very quickly with minimal variability. Conversely, operations that modify the ledger state, such as STORE, UPDATE, and DELETE, exhibit higher execution times. A significant proportion of these operations are completed within the range of approximately 2000 to 2500 ms. This behaviour is due to the RAFT consensus mechanism in Hyperledger Fabric, where a batch window waits for several transactions to fill the block. As these transactions are sent in isolation, the block has to wait for the full window to be committed.

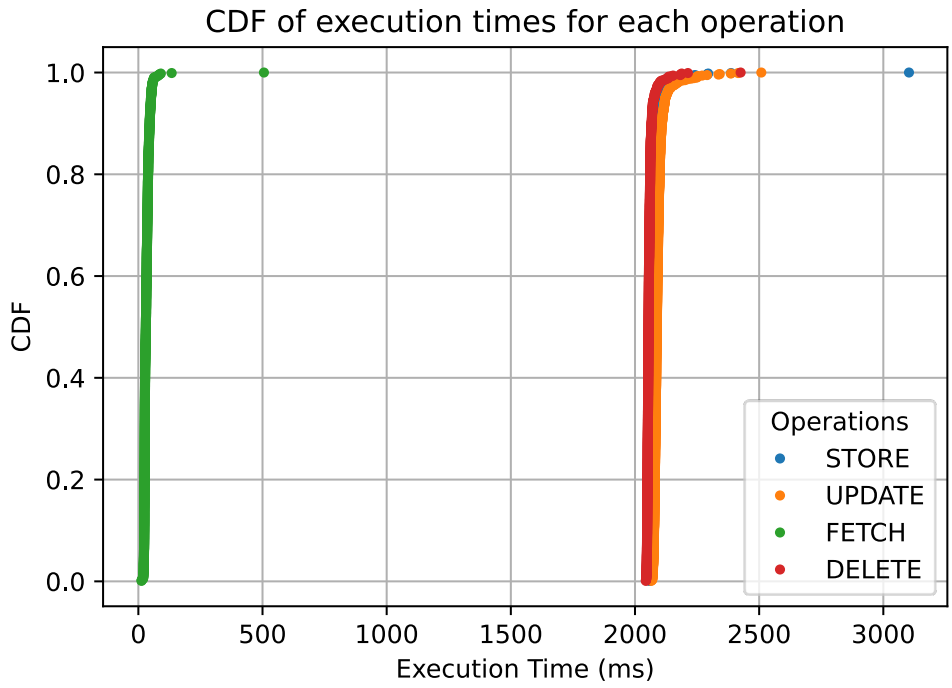

Figure 4-24: CDF of the blockchain operations using topo e2e.

To illustrate the impact of the execution times in mimicking a realistic operation, a throughput test was carried out with its results illustrated in Figure 4-25. During the test, 1000 STORE operations were executed at different transactions per second (TPS) rates for 10TPS, 50TPS, 250TPS, and 500TPS. After execution, the real throughput of the operations was recorded. As showcased in Fig. 6, the execution times significantly influenced the real TPS. At lower TPS rates, the system managed to maintain throughput close to the expected values, indicating efficient handling of operations with minimal delays. However, as the TPS rate increased, the real throughput diverged from the expected values, highlighting the system's struggle to maintain the higher load. This divergence is particularly pronounced at 500TPS.

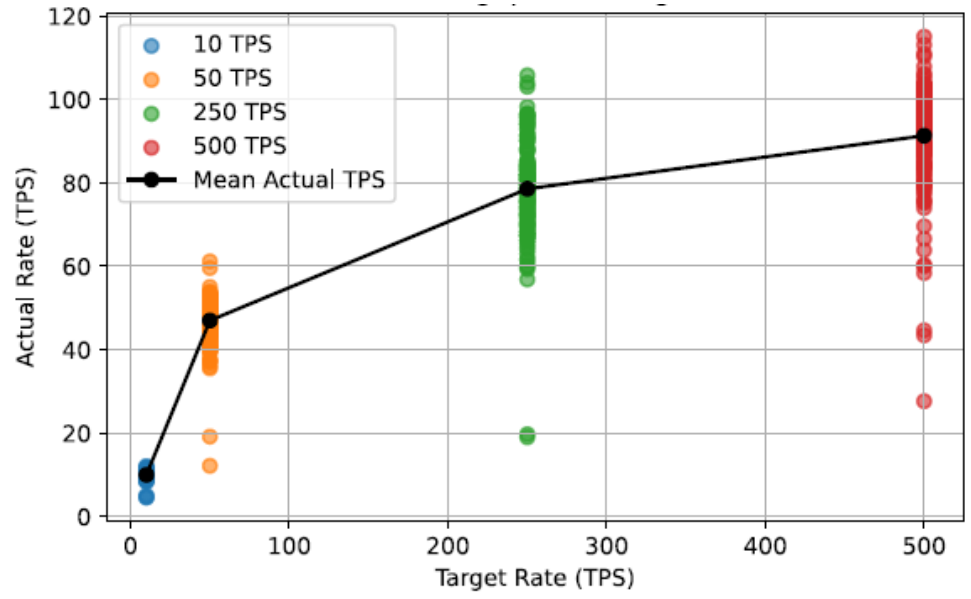

Figure 4-25: Transaction throughput for the STORE operation - topo e2e.

Optimizing the execution times for state-changing operations is crucial for maintaining high performance, especially under heavy transactional loads. This can be achieved by carefully tuning the Hyperledger Fabric block configuration, namely the BatchTimeout and BatchSize, to adjust it to the needs of the expected operation of the system.

4.3.2.1 API for DTL gateway

The primary role of this service is to facilitate interactions with a distributed ledger technology (DLT) gateway, enabling recording, retrieving, subscribing to records, and querying the status and peers within the DLT network. The API is described using Protocol Buffer.

The DltGatewayService comprises three RPC (Remote Procedure Call) methods. The RecordToDlt method takes a DltRecord message as input and returns a DltRecordStatus message, which likely indicates the result of the record operation. The GetFromDlt method accepts a DltRecordId and returns the corresponding DltRecord, enabling retrieval of specific records. The SubscribeToDlt method facilitates real-time updates by allowing clients to subscribe to certain record events, returning a stream of DltRecordEvent messages.

Several enums are defined to categorize types and statuses within the DLT system. The DltRecordTypeEnum enumerates different types of records, such as CONTEXT, TOPOLOGY, DEVICE, LINK, SERVICE, and SLICE, providing a clear classification for record types. The DltRecordOperationEnum lists possible operations on records, including ADD, UPDATE, and DELETE, to specify the action to be performed. The DltRecordStatusEnum defines possible outcomes of record operations, such as SUCCEEDED and FAILED, indicating whether an operation was successful. Additionally, the DltStatusEnum outlines various statuses for the TeraFlow controller, such as NOTAVAILABLE, INITIALIZED, AVAILABLE, and DEINIT, describing the controller's current state.

The protocol buffer also defines several messages to structure the data exchanged within the service. The DltRecordId message includes identifiers for a domain and a specific record within that domain, using UUIDs for uniqueness. The DltRecord message represents a record with an ID, operation type, and JSON-encoded content, encapsulating all necessary details for record manipulation. The DltRecordSubscription message allows clients to specify event types and operations they are interested in, supporting real-time event subscriptions with filtering options. The DltRecordEvent message contains common event data, such as a timestamp and event type, along with the associated record ID, providing context for each event. The DltRecordStatus message includes the status of a record operation and any error messages, offering feedback on the success or failure of operations. The DltPeerStatus message represents the status of a TeraFlow controller instance, while the DltPeerStatusList aggregates multiple DltPeerStatus messages, giving a comprehensive view of peer statuses.

5 Conclusions

5.1 System design validation

This section provides the indication on how the current system design (blueprint in Figure 2-1) fulfils the 6G system design principles set in D2.1 [HEX223-D21] on the basis of the results from the evaluation of the System-PoC #B (section 3.5) and of the security, privacy and system-level resilience (chapter 4).

For each of the design principle discussed below, specific aspects in the 6G system blueprint and the involved system components in the evaluation are detailed. Proof points highlight underscore key results that validate adherence to the design principles. It should be noted that a single system component may contribute to fulfilling multiple design principles. To avoid repetition in proof-point description, highlights focus primarily on involved components where proof-points have not already been reported in other principles.

- **Validation of System design principle 1: Support and exposure of 6G services and capabilities**

Principle purpose:

The design of the 6G system should enable efficient incorporation of 6G digital services, such as sensing and computational offloading. The architecture solution should also be able to expose new and existing capabilities to E2E applications.

Specific aspects and involved components evaluated in this iteration:

The first aspect covers the provision of compute service such as offloading part of application running into the devices to the network (involved System-Poc #B components: Service autoscaling and migration in 3.3.9, trustworthy flexible topologies and beyond communication aspects in 3.3.8).

Another aspect handles the exposure of services to 6G customers within an intent-based approach provides some intent-based interfaces so that 6G customers (e.g. Application developers, enterprises from vertical segments) can request 6G services by an intent expressed with a non-network expert language (System-PoC #B component Intent-based Network solution in 3.3.2).

In-network service exposure is also important with for example the management capabilities exposure framework which offers a service bus for smooth Hexa-X-II management capability interoperability (System-PoC component: Management capabilities exposure framework in 3.3.5)

Proof-point highlights:

Demonstration of compute offloading has been realized for scenario of cobots in a warehouse (see 3.5.3.2, 3.5.3.3) and for AI-assisted E2E lifecycle management of a 6G latency-sensitive service across the cloud continuum (see 3.5.3.4). Both highlight the benefit of hybrid deployment where offloading heavy computational workload at the edge/cloud nodes significantly improve robot application performance and efficiency.

The System-PoC #B demonstrates the role of the Management capabilities exposure framework (MCEF) in providing a communication channel and exposing capabilities within the E2E architecture and to third-party entities. Functional validation is reported in 3.3.5.

System-PoC #B implements DSM-IME components for intent based management and demonstrates a WebUI module permitting the users (e.g. network administrator or verticals) to express different types of requests in natural language. Then the Intent translation and provisioning is using NLP processing to interpret the request. NLP processing time is within a few hundredths of milliseconds. Generated intent and intent report data objects following the 3GPP data model are presented to demonstrate the correct interpretation of the initial human-based request.

- **Validation of System design principle 2: Full automation and optimization**

Principle purpose:

The architecture should support full automation of network and service management operations, utilizing distributed AI/ML agents to manage and optimize the system without human interaction.

Specific aspects and involved components evaluated in this iteration:

For the system design, it requires a system built on a pervasive infrastructure across the compute continuum, to provide seamless service orchestration across diverse scenarios. A multi-platform orchestration should account for components providing synergetic resource orchestration mechanisms for the compute continuum [System-PoC #B component: Service and resource orchestration, described in 3.3.3). Closed loop control is required for real time zero touch automation (System-PoC #B component: Closed loop automation, described in 3.3.4). Distributed AI/ML agents to optimize the system without human interaction is another key component in the design of the system (System-PoC #B component: AI-assisted E2E life-cycle management of a 6G latency-sensitive service across the compute continuum (service autoscaling and migration), described in 3.3.9).

Proof-point highlights:

Zero-touch Cobot-based video surveillance: Most of the time, deployment time from an intent, including NLP processing, feasibility check processing and service instance creation, can be achieved within 250s. The resource orchestration demonstrates functionality to deploy service over a set of cloud resources provided by different providers across the cloud-continuum. It also provisions the close-loop functions to enable service automation. Next, integrated Closed-Loop automation is functionally demonstrated for Cobot Service Migration. Once the battery of the patrolling cobot falls below a certain threshold, a proper closed-loop request the service migration towards another suitable cobot, guaranteeing the service continuity. While measured network latency and throughput fulfil the application needs, work is under progress for reducing the service downtime.

Besides, RL-based joint autoscaling and placement/migration for 6G latency-sensitive services (smart manufacturing services) demonstrates Service Level Objectives (SLOs) satisfaction in terms of E2E latency requirements.

- **Validation of system design principle 3: Flexibility to different network scenarios**

Principle purpose:

The architecture should be designed to support digital inclusion. Addition of service capabilities and new service endpoints can be done at run-time without changes to existing E2E services. The network should support increased application awareness and adaptive QoS/QoE.

Specific aspects and involved components evaluated in this iteration:

Fulfilling this design principle requires establishing a pervasive service management and orchestration framework. Multi-cluster resource orchestration brings flexibility in the deployment of services over multi-cluster environments (System-PoC #B component: Service and resource provisioning in 3.3.3). Federated orchestration components can allow dynamic connections to third party networks capabilities, enhancing flexibility to different network scenarios and topologies (System-PoC component: DLT-based service federation in 3.3.9.2)

Flexible topologies can adapt to traffic bursts and user requirements by network formation with mesh-like structure incorporating UAVs (involved System-PoC #B components: trustworthy flexible topologies and beyond communication aspects in 3.3.8).

Proof-point highlights:

The benefit of introducing a flexible topology has been demonstrated in the cobot-powered warehouse inventory management. It enables task distribution between UAVs and edge computing resources in 6G networks may extend UAV operational periods. UAV-to-edge communication can maintain performance without significantly draining the UAV battery during moderate computational tasks (see results in 3.5.3.1).

Programmable and flexible network configuration the capabilities of the TFS controller has been demonstrate capabilities to seamlessly integrate and manage both the packet and optical layers of transport networks by testing service connectivity provisioning requests. The controller automatically configured the network

elements using the gNMI/OpenConfig protocols for the packet routers, and the Transport API (TAPI) is utilized to configure the optical layer.

- **Validation of system design principle 4: Scalability**

Principle purpose:

The system architecture needs to be scalable to different networks and deployments. The architecture should support very small to very large-scale deployments, by scaling up and down network resources based on mobility and time-varying traffic needs and by utilizing the underlying shared cloud platform.

Specific aspects and involved components evaluated in this iteration:

Fulfilling this design principle focuses on creating a pervasive service management and orchestration system for e.g., scaling up and down based on mobility and time-varying traffic needs (System-PoC component: service autoscaling and migration described in 3.3.9.1).

Components of decentralized orchestrations will support both small and large-scale deployments. (System-PoC #B components: Extreme-edge cluster emulation on high availability scenarios, DLT-based service federation, both described in 3.3.9.2)

Proof-point highlights:

As said earlier, RL-based joint autoscaling and placement/migration for 6G latency-sensitive services have been successfully demonstrated for low latency services in a scenario of smart manufacturing (in 3.3.9). In its current version, the algorithm is reactive, thus showing a delay in optimal decision making when unanticipated workload increases.

DLT-based service federation request triggers DLT federation between different provider and consumer domains when additional resources are needed. They agree on terms using a blockchain smart contract. The functional demonstration elaborated in 3.3.9.2 considered the object detector migration to the provider's Kubernetes cluster. After successful deployment, the provider shares service details, and the consumer reroutes the video stream to the new federated object detector.

- **Validation of system design principle 5: Resilience and availability.**

Principle purpose:

The architecture should allow mobile network operators (MNOs) to build deployments with high resilience and availability. The architecture shall support separation of control plane (CP) and user plane (UP), and resilient mobility solutions as a method to provide service availability. Furthermore, the architecture should support subnetwork resilience e.g., if a subnetwork loses connectivity it should connect with another subnetwork to remove single point of failures.

Specific aspects and involved components evaluated in this iteration:

The 6G E2E system requires pervasive service management and orchestration that provide auto-scaling mechanisms to adapt to traffic to ensure continuous and reliable services (System-PoC component: service autoscaling and migration described in 3.3.9.1). Closed loop automation can also help in adapting resource or migrate the service to guarantee service continuity. Flexible network topologies can also help to improve availability and reliability of the network (System-PoC #B component: trustworthy flexible topologies described in 3.3.8). E2E resilience simulation tool described in 4.2.3 can also be used in the design phase for testing several traffic scenarios can be run to check if the dimensioning and auto-scaling mechanisms are enough to face the traffic increase or if additional resources are required.

Proof-point highlights:

High availability in an emulated extreme-edge infrastructure was demonstrated in 3.4.2 for real-time video streaming service. The volatile extreme-edge nodes unexpectedly connect and disconnect, but the service remains operational. Additional replicas deploy on remaining nodes, and the orchestrator redeploy missing replicas as needed.

E2E resilience simulation reported in 4.2.3 validates the capability to evaluate the VNF availability and how it is impacted by the frequency of health check. It has also been successfully used to evaluate a given network service chain ability to reach the network availability above 5 nines required in a vertical use case (tele-action use case for remote decoupling protections for the distributed generation power stations in the electric grid).

- **Validation of system design principle 6: Persistent security and privacy**

Principle purpose:

The 6G system should be able to address current as well as future threats in a resilient manner and incorporate security fundamentals in its design. Furthermore, the 6G system should inherently support the preservation of privacy and allow different levels of anonymity also for future services.

Specific aspects and involved components evaluated in this iteration:

The objective is to evaluate a comprehensive framework in the 6G E2E system that provide controls to ensure security and privacy against the specific 6G threats identified in D2.2 [HEX223 D2.2]. It covers the integration of some of security and privacy fundamentals studied in D2.3 [HEX224 D2.3] covering physical layer security, trustworthy AI, Quantum resistant cryptography, Level of Trust assessment, Confidential Network deployment, Distributed ledgers. Those components are described in chapter 4.

Proof-point highlights:

The evaluation results provided in chapter 4 provide evidence on the feasibility of the enablers, and associated SPR controls, proposed in previous deliverables. These results are also useful to evaluate their impact on network features and performance. These results are provided according to the planned TRL for each proposed enabler.

For those enablers at basic TRLs, the outcomes of conceptual evaluation are provided, as results of simulation or laboratory experiments, together with relevant progress on the concepts themselves, as theoretical and experimental evidence is gathered.

SPR enablers at validation TRLs are validated in experimental environments devised and applied to evaluate the supporting SPR controls and the corresponding enabler interfaces, with details of the performed experiments and the available measurements, together with the foreseen characteristics of the interfaces for using these enablers or directly interacting with the corresponding controls.

For enablers at the highest TRLs, those associated with development stages to become part of the 6G security toolbox, the reported results come from initial experiments evaluating the conditions and cases for the application of these technologies, as well as the information model to be applied in the specific control interfaces, and considerations on their use.

- **Validation of system design principle 7: Internal interfaces are cloud optimized.**

Principle purpose:

Network interfaces should be designed to be used in a cloud-native environment, utilizing state-of-the-art cloud technologies, design patterns, platforms and IT tools in a coherent and consistent manner. Care should be taken to design proper service separation to maximize the potential for service reuse and allow for service innovation with minimal integration efforts (plug-and-play).

Specific aspects and involved components evaluated in this iteration

For the design of the system, it involves having a cloud-native approach in services development and orchestration. In particular, it involves providing a communication channel between the micro-services of network services and applications, even when those micro-services are deployed across multiple clouds (System-PoC component: Inter-cluster communication described in 3.3.9.4). In addition, another key enabler is a communication channel between management services as well as for exposing those services to third party entities (System-PoC component: Management capabilities exposure framework described in 3.3.5).

Proof-point highlights:

Two alternative options are still explored regarding inter-cluster communication in 3.3.9.4. First, the karmada multi-cluster management tool, and second, Network Service Mesh (NSM).

Functional validation for the management capabilities exposure framework is reported in 3.3.5.

- **Validation of system design principle 10: Minimize environmental footprint and enabling sustainable use cases.**

Principle purpose:

The design of 6G system should ensure that the environmental footprints can be justified, and any increased footprint should be clearly motivated with added value, cost efficiency, and societal benefits. The architecture design should enable to disable or deactivate functions that are not in use, either short-term or long-term, to ensure zero power at zero load. It should also facilitate circular practices through higher modularity of components to limit the replacement to only the degraded part or compatibility with software update to extend the lifetime.

Specific aspects and involved components evaluated in this iteration:

E2E orchestration must facilitate energy-efficient and cost-conscious operations. It should enable energy-aware placement algorithms as well as energy-efficient offloading of compute across the cloud continuum (System-PoC #B component: Functionality Allocation function in Trustworthy flexible topologies and beyond communication aspects). Besides, it implies efficient means to monitor energy resource consumption from end devices, network and compute nodes, including energy consumed by AI models training and inference.

Proof-point highlights:

In cobot-powered warehouse inventory management scenario, the Functionality Allocation provides energy consumption gains compared to the round-robin placement between 9.9% and 50.9% depending on the scarcity of compute resource.

5.2 Conclusion and next steps

The Hexa-X-II project represents a pivotal step towards the realization of 6G networks, aiming to deliver a comprehensive 6G E2E system evaluation. This document encapsulates the outcomes of the second iteration of the System-PoC #B and provides valuable insights into the design, implementation, and evaluation of 6G systems. The findings presented herein underscore the project's commitment to addressing the multifaceted challenges of future communication systems through innovative approaches in system design, management, orchestration, and sustainability.

System Design Validation

The current system design for 6G (chapter 2), as outlined in the Hexa-X-II blueprint, emphasizes several key principles, including full automation, scalability, resilience, and security. The integration of intent-based network solutions (subsection 3.3.2), closed-loop automation (subsection 3.3.4), and AI-assisted lifecycle management (subsection 3.3.9) demonstrates significant advancements in the management and orchestration of 6G services. These components collectively ensure that the system can support a wide range of network scenarios, optimize resource usage, and maintain high levels of performance and availability.

The system design validation involved rigorous testing of various components, such as the data fusion/telemetry, programmable network configurations (subsection 3.3.6), and trustworthy flexible topologies (subsection 3.3.8). These elements are crucial for achieving the desired levels of flexibility, scalability, and resilience in 6G networks. As presented more in detail in the previous section, the evaluation results confirm that the current system design principles set forth in D2.1 [HEX223-D21] are effectively met, providing a robust foundation for future 6G developments.

Performance and Key Value Indicators

System-PoC #B's evaluation results highlight the system's ability to meet the stringent KPIs and KVIs essential for 6G networks (subsections 3.5.1, 3.5.2). Notably, the focus on latency-sensitive services, such as real-time video surveillance using cobots, underscores the system's capability to handle compute-intensive operations with ultra-low latency. The deployment of intent-based service provisioning and closed-loop automation ensures zero-downtime and optimal resource utilization, thereby enhancing the overall system performance and reliability (subsection 3.5.3).

Moreover, the assessment of social, environmental, and economic sustainability aspects within System-PoC #B demonstrates the project's holistic approach to 6G development. The incorporation of energy-efficient operation policies, resilient network topologies, and reduced operational costs aligns with the sustainability goals of minimizing environmental footprints and enabling long-term economic viability.

Security and Privacy

Persistent security and privacy (chapter 4) are paramount in the design of the 6G E2E system. The integration of comprehensive security frameworks (subsections 4.1.1 - 4.1.4), including quantum-resistant cryptography (subsection 4.2.2), trustworthy AI (subsection 4.2.1), and confidential network deployment (subsection 4.3.1), ensures a secure and trustworthy environment for 6G services. The project's focus on addressing current and future threats through inherent design principles and advanced security measures signifies a proactive approach to safeguarding 6G networks.

Future Directions

As the Hexa-X-II project progresses towards its final reporting period, the outcomes from this deliverable will serve as a crucial reference for the final round of WP2 deliverables and further discussions with other SNS projects. The insights gained from System-PoC #B will allow to follow improving and refining of the 6G system designs, ensuring that future iterations are well-equipped to meet the evolving demands of next-generation communication networks.

Regarding the System-PoC, the results presented in this deliverable will trigger the last phase (I.e., System-PoC #C) with new results from the current enablers and solutions and with the addition of other technologies such as radio communications.

In conclusion, the Hexa-X-II project is making significant strides in advancing 6G technologies through its comprehensive system evaluation and innovative design principles. The validation of system components against stringent KPIs and KVIs, coupled with a strong focus on security, privacy, and sustainability, positions the project as a cornerstone in the journey towards realizing the full potential of 6G networks. As we move forward, the collaborative efforts within the Hexa-X-II consortium and the broader SNS community will be instrumental in shaping the future of global communication systems.

6 References

[28.312] 3GPP TS 28.312, “Management and orchestration; Intent driven management service for mobile networks,” v18.2.1, December 2023.

[38.401] 3GPP, “NG-RAN Architecture Description,” 3GPP TS 38.401 version 18.1.0 Release 18 March 2024.

[38.801] 3GPP, “Study on new radio access technology: Radio access architecture and interfaces,” 3GPP TR 38.801 V14.0.0, Release 14, March 2017.

[5GE18-D11] 5G-EVE Deliverable D1.1 "Requirements Definition & Analysis from Participant Vertical Industries," October 2018. [Online]. Available: https://doi.org/10.5281/zenodo.3530391

[HEX2-AAG+23] P. Alemany, D. Adanza, L. Grifre, R. Vilalta, and R. Muñoz, “Grouping Intent-based Packet-Optical Connectivity Services,” 49th European Conference on Optical Communications (ECOC 2023), Glasgow, Scotland, October 2023.

[ACR24] ACROSS SNS JU Project. Availabe at: https://across-he.eu/

[BJP+21] L. Bonati, P. Johari, M. Polese, S. D’Oro, S. Mohanti, M. Tehrani-Moayyed, D. Villa, S. Shrivastava, C. Tassie, K. Yoder, A. Bagga, P. Patel, V. Petkov, M. Seltser, F Restuccia, A. Gosain, K. R. Chowdhury, S. Basagni, and T. Melodia, “Colosseum: Large-scale wireless experimentation through hardware-in-the-loop network emulation,” in IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN), 2021, pp. 105–113

[CHZ+24] W. Chen, B. Han, Y. Zhu, A. Schmeink, G. Caire, and H. D. Schotten, “Physical layer deception with non-orthogonal multiplexing,” arXiv preprint arXiv:2407.00750 (2024).

[DAK+21] L. Diez, A. M. Alba, W. Kellerer and R. Agüero, "Flexible Functional Split and Fronthaul Delay: A Queuing-Based Model," in IEEE Access, vol. 9, pp. 151049-151066, 2021, doi: 10.1109/ACCESS.2021.3124374.

[HEX2-DUN+24] P. Dass, S. Ujjwal, J. Novotny, Y. Zolotavkin, Z. Laroussi, and S. Köpsell, "Addressing Privacy Concerns in Joint Communication and Sensing for 6G Networks: Challenges and Prospects", arXiv preprint arXiv:2405.01742, 2024.

[ESA23] European space agency (ESA), “5G-HOSTS-SAT 5G hub over-the-air vertical segment validations,” 2023. [Online]. Available: https://connectivity.esa.int/projects/5ghostssat

[GNMI] OpenConfig – gRPC Network Management Interface (gNMI) specification, available at: https://openconfig.net/docs/gnmi/gnmi-specification/

[HCA+22] J. Hoydis, S. Cammerer, F. A. Aoudia, A. Vem, N. Binder, G. Marcus, and A. Keller, “Sionna: An Open-Source Library for Next-Generation Physical Layer Research.” arXiv, 2022. doi: 10.48550/ARXIV.2203.11854.

[HEX223-D21] Hexa-X-II, “Deliverable D2.1: Draft foundation for 6G system design,” June 2023. [Online]. Available: https://hexa-x-ii.eu/https://hexa-x-ii.eu/wp-content/uploads/2023/07/Hexa-X-II_D2.1_web.pdf

[HEX223-D22] Hexa-X-II, “Deliverable D2.2: Foundation of overall 6G system design and preliminary evaluation results,” December 2023. [Online]. Available: https ://hexa-x-ii.eu/wp-content/uploads/2024/01/Hexa-X-II_D2.2_FINAL.pdf

[HEX224-D13] Hexa-X-II, “Deliverable D1.3: Environmental and social view on 6G” March 2023. [Online]. Available: https://hexa-x-ii.eu/wp-content/uploads/2024/03/Hexa-X-II_D1.3_v1.00_GA_approved.pdf

[HEX224-D23] Hexa-X-II, “Deliverable D2.3: Interim overall 6G system,” June 2024. [Online]. Available: https://hexa-x-ii.eu/

[HEX224-D33] Hexa-X-II project, Deliverable D3.3, "Initial analysis of architectural enablers and framework," April 2024, https://hexa-x-ii.eu/wp-content/uploads/2024/04/Hexa-X-II_D3.3_v1.0.pdf

[HEX224-D34] Hexa-X-II project, Deliverable D3.4, "Preview of the final architectural framework and analysis," December 2024, to be published in https://hexa-x-ii.eu/.

[HEX225-D35] Hexa-X-II project, Deliverable D3.5, "Final architectural framework and analysis," February 2025, to be published in https://hexa-x-ii.eu/.

[HEX224-D63] Hexa-X-II, "Deliverable D6.3: Initial Design of 6G Smart Network Management Framework", June 2024. Available at https://hexa-x-ii.eu/wp-content/uploads/2024/07/Hexa-X-II_D6-3_v1.0.pdf

[HEX23-D53] Hexa-X project, Deliverable D5.3, "Final 6G architectural enablers and technological solutions," April 2023, available at https://hexa-x-ii.eu/

[HEX2-HZS+23] B. Han, Y. Zhu, A. Schmeink, and H. Schotten, "Non-orthogonal multiplexing in the FBL regime enhances physical layer security with deception," In 2023 IEEE 24th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), pp. 211-215, September 2023.

[HEX2-KPK+24] F. Karakoç, B.G. Paltun, L. Karaçay, Ö. Tuna, R. Fuladi, U. Gülen, "FASIL: A challenge-based framework for secure and privacy-preserving federated learning", IACR ePrint Archive 2024/1028 https://eprint.iacr.org/2024/1028, 2024.

[HEX2-MMC+23] A. Mayya, M. Mitev, A. Chorti, and G. Fettweis, "A SKG Security Challenge: Indoor SKG Under an On-The-Shoulder Eavesdropping Attack", In the proceedings of IEEE GLOBECOM 2023, pp. 3451-3456.

[HEX2-RAP+24] Y. Rumesh, D. Attanayaka, P. Porambage, J. Pinola, J. Groen, and K. Chowdhury, "Federated Learning for Anomaly Detection in Open RAN: Security Architecture Within a Digital Twin", In the proceedings of European Conference on Networks and Communications (EuCNC) and 6G Summit, 2024.

[HEX2-VAM+23] R. Vilalta, D. Adanza, C. Manso, P. Alemany, L. Gifre, R.Casellas, R. Martínez, and R. Muñoz. "Demonstration of Intent-Based Networking for End-to-End Packet Optical Cloud-native SDN Orchestration," 49th European Conference on Optical Communications (ECOC 2023), October 2023, Glasgow, Scotland. https://doi.org/10.5281/zenodo.10286071

[Int18] Intel, Whitepaper, "Performance Considerations for Intel Software Guard Extensions (Intel SGX) Applications, 2018, https://cdrdv2-public.intel.com/671502/intel-sgx-performance-considerations.pdf

[Kai23] Kaitotek, "Measuring QoS", 2023 [Online]. Available: https://www.kaitotek.com/qosium/measuring

[KBS+24] A. Krause, F. Burmeister, P. Schulz, M. D. Khursheed, S. Ma, and G. Fettweis, "Advancing Spectrum Anomaly Detection through Digital Twins." TechRxiv, 2024. doi: 10.36227/techrxiv.171470424.41432460/v1.

[LBB+98] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition", Proc. IEEE vol. 86, pp. 2278-2324, 1998.

[LCC19] L. M. P. Larsen, A. Checko, and H. L. Christiansen, "A Survey of the Functional Splits Proposed for 5G Mobile Crosshaul Networks," in IEEE Communications Surveys & Tutorials, vol. 21, no. 1, pp. 146-172, Firstquarter 2019, doi: 10.1109/COMST.2018.2868805.

[LDB+22] R. Li, B. Decocq, A. Barros, Y. Fang, and Z. Zeng. "Petri Net-Based Model for 5G and Beyond Networks Resilience Evaluation". In 2022 25th Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), pages 131–135, Paris, France, March 2022.

[LL19] Y. Li and Y. Lu, “LSTM-BA: DDoS detection approach combining LSTM and Bayes,” in IEEE international conference on advanced cloud and big data (CBD), pp. 180–185, 2019.

[MNC+17] R. Muñoz, L. Nadal, R. Casellas, M. S. Moreolo, R. Vilalta, J. M. Fàbrega, R. Martínez, A. Mayoral, F. J. Vílchez, "The ADRENALINE testbed: An SDN/NFV packet/optical transport network and edge/core cloud platform for end-to-end 5G and IoT services," 2017 European Conference on Networks and Communications (EuCNC), Oulu, Finland, 2017, pp. 1-5, doi: 10.1109/EuCNC.2017.7980775.

[MRT23] N. Malm, K. Ruttik, and O. Tirkkonen, “Midhaul Performance Modelling Using Commodity Hardware C-RAN Testbed,” Wireless Personal Communications, 131:1339–1363, April 2023.

[NISTZT20] NIST, “SP 800-207. Zero Trust Architecture”, https://csrc.nist.gov/pubs/sp/800/207/final

[OAI24] Open Air Interface Community webpage, https://openairinterface.org/

[OAI-COR] Open Air Interface, “Openairinterface 5G Core Network Implementation,” open-source codes available at: https://gitlab.eurecom.fr/oai/cn5g

[OAI-RAN] Open Air Interface, “Openairinterface 5G Wireless implementation,” open-source codes available at: https://gitlab.eurecom.fr/oai/openairinterface5g

[OAI-RF] OAI, ”RF simulator,” GITLAB README file, https://gitlab.eurecom.fr/oai/openairinterface5g/-/blob/develop/radio/rfsimulator/README.md

[OAPI24] OpenAPI Specification, version 3.1.0. Available at: https://swagger.io/specification/

[ONF-TAPI] GitHub – ONF Transport API 2.4.0. https://github.com/Open-Network-Models-and-Interfaces-ONMI/TAPI/releases/tag/v2.4.0

[OPE24] OpenConfig – Data models. Available at: https://openconfig.net/projects/models/

[OQS24] Open Quantum Safe (OQS) project, https://openquantumsafe.org

[ORA24] Open RAN Alliance webpage, https://www.o-ran.org/

[ORCUS24] O-RAN WG4, “O-RAN Control, User and Synchronization Plane Specification 14.0,” O-RAN specification, O-RAN.WG4.CUS.0-R003-v14.00, February 2024.

[PSG+23] J. Pérez-Romero, O. Sallent, A. Gelonch, X. Gelabert, B. Klaiqi, M. Kahn, and D. Campoy, "A Tutorial on the Characterisation and Modelling of Low Layer Functional Splits for Flexible Radio Access Networks in 5G and Beyond," in IEEE Communications Surveys & Tutorials, vol. 25, no. 4, pp. 2791-2833, Fourthquarter 2023, doi: 10.1109/COMST.2023.3296821.

[Quj24] Qujata. Next Gen Crypto. Quantified, https://github.com/att/qujata/blob/main/README.md#overview

[SPQR24] SPQR Cluster for EU Transition to Post-Quantum Cryptography, https://pqreact.eu/spqr-cluster/

[TFS24] ETSI GitLab TeraFlowSDN – Wiki pages. https://labs.etsi.org/rep/tfs/controller/-/wikis/home

[TMF63321] TMForum, “TMF633 Service Catalog API User Guide v4.0.0”, https://www.tmforum.org/resources/standard/tmf633-service-catalog-api-user-guide-v4-0-0/

[TMF64121] TMForum, “TMF641 Service Ordering Management API User Guide v4.1.1”, https://www.tmforum.org/resources/standard/tmf641-service-ordering-management-api-user-guide-v4-1-1/

[TMF65321] TMForum, “TMF653 Service Test Management API User Guide v4.1.0”, https://www.tmforum.org/resources/standard/tmf653-service-test-management-api-user-guide-v4-1-0/

[VEP+24] J. Velazquez, M. Elorza, A. Pastor, D. Lopez, and J. A. Alonso, “Implementation of a traffic flow path verification system in a data network”, 2024 European Conference on Networks

and Communications & 6G Summit (EuCNC/6G Summit), June 2024, doi: 10.1109/EuCNC/6GSummit60053.2024.10597042

[WGM+23] T. Wild, A. Grudnitsky, S. Mandelli, M. Henninger, J. Guan, and F. Schaich, "6G integrated sensing and communication: From vision to realization", In the proceedings of 20th European Radar Conference (EuRAD). pp. 355–358, 2023.

[WKS21] P. Walther, R. Knauer, and T. Strufe, "Ultra-Wideband Channel State Information and Localization for Physical Layer Security", IEEE Dataport, February 16, 2021, DOI: https://dx.doi.org/10.21227/0wej-bc28.

[ZLH19] L. Zhu, Z. Liu, and S. Han, "Deep Leakage from Gradients", Neural Information Processing Systems, 2019.

ANNEX A

A.1 Logs related to the intent-based deployment and provisioning.

***** INTENT REQUEST DEPLOYMENT *****
| Trigger intent deployment -> 2024-05-23 07:23:17.138564
| Sending intent to be interpreted.
| | Recognize Operation.
| | Recognize Resources.
| | Recognize Location.
| | Recognize Recursivity.
| | NLP applied.
| Intent stored in DB -> 2024-05-23 07:23:17.152287
| The intent fulfillment received the resources gathered on start-up.
| Intent Feasibility done -> 2024-05-23 07:23:17.152665
| Intent Report Creation.
| | Main section added.
| | Feasibility section added.
| Intent expectations deployment.
| | Activating intent.
| | Expectation identified and processing:
| | | intentExpectations: {“expectation_id”: 9a294312-ed73-4765-b473-178494c08c34, …}
| | | Selected Service Resource:
| | | | Name: cobot streaming-app blueprint
| | | | ID: 2c5b3484-35ff-4c47-9cd3-0de9049deecd
| | Requesting service to SOCCER.
| | The SLA policy for resilience has been identified.
| | | Requested service
| | | Service ID: 2c5b3484-35ff-4c47-9cd3-0de9049deecd
| | | Service Name: cobot streaming-app blueprint
| | | Service Description: cobot streaming application service blueprint.
| | | Service Config: {…}
| | Create SOCCER service instance: {'id': '57dc6cd4-3a56-4782-84ed-f9d702949801', …}
| | Instantiate SOCCER vertical services: 57dc6cd4-3a56-4782-84ed-f9d702949801
| | Waits until SOCCER vertical service is instantiated: {'id': '57dc6cd4-3a56-4782-84ed-…}
| | SOCCER vertical service is INSTANTIATED.
| | The inclusion of fulfilment report values will be carried out for this expectation object.
| | The fulfilment report data has been updated on the database.
| Intent deployed -> 2024-05-23 07:23:41.969829
***** INTENT REQUEST DEPLOYMENT COMPLETED *****
INFO: 192.168.125.222:64174 - "POST /ibn-ime/cttc/1.0/intent/intent%3D0?stringRequest=create%20a%20cobot%20application%20on%20warehouse%2024%20with%20zerodowntime HTTP/1.1" 200 OK

Figure A-1: Intent provisioning requests logs example.

```
*** INTENT QUERY ***
| Retrieving intent with id: ebb3c42e-ef5b-4588-a550-7bbe3c4412f7
| The manager will connect to the database to query all the intents.
| The object response will be sent straight away.
INFO:     192.168.125.222:64225 - "GET /ibn-ime/cttc/1.0/intent/intent%3Debb3c42e-ef5b-4588-a550-7bbe3c4412f7 HTTP/1.1" 200 OK
```

Figure A-2: Intent retrieve request logs example.

```
*** INTENT REPORT QUERY ***
| Retrieving intent report with id: c10d602f-27a3-41a1-80ee-32ee7dd9fff3
| The manager will connect to the database to query all the intent reports.
| The object response will be sent straight away.
```

Figure A-3: Intent report retrieve request logs example.

Table A-1: Intent deployment time samples

Request number	NLP (sec)	Feasibility (sec)	created the vertical instance (sec)
1	0.1169	0.0005	1308.0807
2	0.0214	0.0007	151.3584
3	0.0114	0.0004	1673.5246
4	0.0137	0.0004	576.0144
5	0.0181	0.0005	23.2649
6	0.0231	0.0004	1525.0907
7	0.0276	0.0010	20.2775
8	0.0321	0.0006	20.1229
9	0.0214	0.0007	14.4090
10	0.0114	0.0004	27.510
11	0.0137	0.0004	24.8171
12	0.0148	0.0004	23.2265
13	0.0181	0.0005	23.2649
14	0.0231	0.0004	25.0907
15	0.0276	0.0010	20.2775