Call: H2020-ICT-2020-2
Project reference: 101015956

Project Name:
A flagship for B5G/6G vision and intelligent fabric of technology enablers connecting human, physical, and digital worlds
Hexa-X

Deliverable D6.3 Final evaluation of service management and orchestration mechanisms

Date of delivery:	30/04/2023	Version:	1.1
Start date of project:	01/01/2021	Duration:	30 months

Document properties:

Document Number:	D6.3
Document Title:	Final evaluation of service management and orchestration mechanisms.
Editor(s):	Sławomir Kukliński (ORA)
Authors:	Sławomir Kukliński (ORA), Rafał Tępiński (ORA), Gregorio Martínez Pérez (UMU), Manuel Gil Pérez (UMU), José María Jorquera Valero (UMU), Mohammad Asif Habibi (TUK), Giada Landi (NXW), Giacomo Bernini (NXW), Erin Elizabeth Seder (NXW), Pietro Piscione (NXW), Michael De Angelis (NXW), Jorge Martín (UC3), Jesús Pérez (UC3), Pablo Serrano (UC3), Antonio Virdis (UPI), Ignacio Labrador Pavón (ATO), Adrián Gallego (ATO), Enrique Lluesma Marti (ATO), Bessem Sayadi (NOK-FR), Frédéric Faucheux (NOK-FR), Cedric Morin (BCO), Cao-Thanh Phan (BCO), Christos Ntogkas (WIN), Ioannis Prodromos Belikaidis (WIN)
Contractual Date of Delivery:	30/04/2023
Dissemination level:	Public
Status:	Final
Version:	1.1
File Name:	Hexa-X_D6.3_1.1.pdf

Revision History

Revision	Date	Issued by	Description
V1.0	02.04.2023	HEXA-X WP6	Version for the General Assembly (GA) review
V1.1	28.04.2023	HEXA-X WP6	Includes the updates requested from the GA

Abstract

This Deliverable D6.3 contains the final evaluation of management and orchestration (M&O) mechanisms of the Hexa-X project and concludes work in WP6, which addresses. This document is built upon previous deliverable D6.2, which described the architectural design of these mechanisms. It demonstrates the selected M&O mechanisms in the form of two demos in alignment with selected Hexa-X Use Cases. The final assessment is performed through measurements of improvements in areas such as energy efficiency, intelligent network reconfiguration, onboarding time and service continuity.

Keywords

B5G, 6G, M&O, MANO, OAM, management and orchestration, AI/ML-based service and network management, continuum management and orchestration, extreme-edge, cloud-native, SBMA, SBA, architectural design.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101015956.

Executive Summary

This report is the third deliverable of the Hexa-X Work Package 6 (WP6). It presents the evaluation of service management and orchestration (M&O) mechanisms for Hexa-X. The deliverable reflects the work done in WP6 from month 17 (May 2022) until month 28 (April 2023). This document introduces the implementation of novelties described in the previous Deliverable D6.2 [HEX22-D62], such as (1) unified orchestration across the "extreme-edge, edge, core" continuum, (2) unified management and orchestration across multiple domains owned and administered by different stakeholders, (3) increasing levels of automation, (4) adoption of data-driven and AI/ML techniques in the M&O system and (5) adoption of the cloud-native principles in the telco-grade environment.

The main part of this deliverable focuses on the description of two demos (Demo #4, Demo #5) and complementary lab experiments. Each of the demos is presented uniformly, consisting of (i) *Demo overview*, (ii) *Innovations related to the demo* and (iii) *Demo implementation*, of which the last one additionally describes individual scenarios related to a particular demo. All of the scenarios are again presented in a uniform fashion, consisting of (a) *scenario description*, (b) *software components*, (c) *functional behaviour* and (d) *deployment.*

Demo #4, "Handling unexpected situations in industrial contexts", consists of a set of three mobile robots in a simulated industrial environment, which are able to work as Digital Twins. The target of this objective is to turn AI/ML into an essential component of B5G/6G technology. In order to address this objective, Demo #4 presents three scenarios:

- Scenario 4.1 "Continuum (cloud, edge, extreme-edge) M&O of a Digital Twins service".
- Scenario 4.2 "Handling unexpected events using functions placement".
- Scenario 4.3 "Improving service downtime and reducing costs using predictive orchestration".

These scenarios build upon each other in order to show how distinct enablers can be used together to accomplish the targets of related objectives.

On the other hand, Demo #5, "Data-driven device-edge-cloud continuum management", presents four scenarios which, unlike scenarios in Demo #4, focus on specific facets each:

- Scenario 5.1, "Continuum orchestration of AI/ML-Driven Traffic Lights Control Service", aims at demonstrating how AI/ML-Driven approach can improve road traffic mobility compared to the legacy non-AI/ML approach.
- Scenario 5.2, "Prediction-based URLLC service orchestration and optimization", aims at demonstrating how the predictive approach for resource allocation differs from reactive methods.
- Scenario 5.3, "Reactive security for the edge", aims at demonstrating the proposed network management architecture's effectiveness in handling cyber security threats against a vulnerable application deployed at the extreme-edge.
- Scenario 5.4, "MLOps techniques to deploy AI/ML service components", aims to demonstrate the implementation of the MLOps practices in a telecommunications environment by proposing an architecture in which vendor and operator coexist, and communication between them is established.

Complementary lab experiments aim to independently address the topics of network energy efficiency, extreme-edge nodes discovery, as well as the impact of B5G/6G RAN on Scenario 5.1.

In the second part of this document, the evaluation of the proposed service M&O mechanisms is performed. The evaluation includes the following:

- The WP6 contribution to the overall Hexa-X objectives, focusing on Objective 3 (which is the one related to this WP6 according to the Hexa-X workplan), and considering the main outputs towards that objective, the main measurable results, and the evaluation of the quantifiable targets.

- The evaluation of the Hexa-X M&O architectural design, provided in the previous Deliverable D6.2 [HEX22-D62], by means of the demos presented in this deliverable (Demo #4 and Demo #5).
- The evaluation of those KPIs, KVIs and Core Capabilities identified as the most relevant in the M&O scope also in the previous [HEX22-D62], and that have been evaluated in the context of the demos and the complementary experiments in this document.
- A summary of the main lessons learnt.
- Finally, some hints and suggestions for future work.

Table of Contents

List of Figures

List of Tables

1 Scope

This report is the third deliverable of the Hexa-X Work Package 6 (WP6) and presents the evaluation of service management and orchestration (M&O) mechanisms of the M&O architectural framework provided by the Hexa-X project. The selected evaluation mechanisms are linked with the key novelties of the proposed framework and include predictive, data-driven orchestration, continuum orchestration (till the extreme-edge) and adoption of cloud-native principles in the telco-grade environment. The M&O operations of the mentioned framework deeply use AI/ML techniques. All the mechanisms contribute to intelligent orchestration and service management needed for 6G networks. The document evaluates the benefits of the proposed framework and AI-driven mechanisms and their contribution to project quantifiable targets.

2 Abbreviations

2D	Two-dimensional
3D	Three-dimensional
3GPP	3rd Generation Partnership Project
5G	Fifth generation of mobile telecommunications technology.
6G	Sixth generation of mobile telecommunications technology.
ACK	Acknowledgement
AGV	Automated Guided Vehicle
AI	Artificial Intelligence
AL	Abstraction Layer
API	Application Programming Interface
ARIMA	Autoregressive Integrated Moving Average
AS	Access Stratum
B5G	Beyond 5G
BMU	Best Matching Unit
BRMS	Business Rules Management System
CD	Continuous Delivery
CDN	Content Delivery Network
CI	Continuous Integration
CO_2	Carbon Dioxide
CNF	Container Network Functions
CPaaS	Communications Platform as a Service
CP	Control Plane
CPU	Central Processing Unit
CQI	Channel Quality Indicator
CRUD	Create Read Update Delete
CSMF	Communication Service Management Function
CVSS	Common Vulnerability Scoring System
DB	Database
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DevOps	Development and Operations
DoS	Denial of Service

DDoS	Distributed Denial of Service
DRT	Delta Reconfiguration Time
DT	Digital Twins
E2E	End-to-End
EC	Edge Cluster
EEC	Extreme-edge Cluster
ETSI	European Telecommunications Standards Institute
gNB	Next-Generation NodeB
GPIO	General Purpose Input/Output
GPRS	General Packet Radio Service
GTP	GPRS Tunnelling Protocol
GUI	Graphical User Interface
HMI	Human-Machine Interaction
I/O	Input/Output
IDS	Intrusion Detection System
IoT	Internet of Things
IP	Internet Protocol
IPS	Intrusion Prevention System
ISO	International Organization for Standardization
K3s	Lightweight Kubernetes
K8s	Kubernetes
KNF	Kubernetes-Based Virtual Network Function
KPI	Key Performance Indicator
KVI	Key Value Indicator
LAD	Lane Area Detector
LCM	Life Cycle Management
LED	Light Emitting Diode
LIDAR	Light Detecting and Ranging
LoT	Level of Trust
LoTAF	Level of Trust Assessment Function
LSTM	Long Short-Term Memory
M&O	Management and Orchestration
MANO	Management and Orchestration
MA	Micro-aggregation
MAC	Media Access Control
MaaS	Monitoring as a Service
MEC	Mobile Edge Computing
ML	Machine Learning
MLOps	Machine Learning Operations
MNO	Mobile Network Operator
MTTD	Mean Time To Detect
MTTR	Mean Time To Respond
NACK	Negative Acknowledgement

NAS	Non Access Stratum
NF	Network Function
NN	Neural Network
NR	New Radio
NSMF	Network Slice Management Function
NSSMF	Network Slice Subnet Management Function
OAM	Operations Administration and Management
OMNeT	Objective Modular Network Testbed
OS	Operating System
OSM	Open-Source MANO
PMx	Particulate Matter
PGW	Packet Data Network GateWay
PoC	Proof of Concept
QoS	Quality of Service
RADAR	Radio Detecting and Ranging
RAM	Random Access Memory
RAN	Radio Access Network
RCA	Root Cause Analysis
RD	Resource Discovery
REC-EXEC	REsource orchestrator for Continuum across EXtreme-edge, Edge, Cloud
RI	Resource Inventory
RL	Reinforcement Learning
RMSE	Root Mean Square Error
RSU	Road Side Unit
SARIMA	Seasonal AutoRegressive Integrated Moving Average
SBA	Service-Based Architecture
SBMA	Service-Based Management Architecture
SD	Service Deployer
SOM	Self-Organizing Maps
SSH	Secure Shell
SUMO	Simulation of Urban Mobility
SW	Software
TCP	Transmission Control Protocol
TFX	TensorFlow Extended
TL	Traffic Light
TRL	Technology Readiness Level
ToD	Tele-operated Driving
TraCI	Transport Control Interface
TSDB	Time-Series Database
UDP	User Datagram Protocol
UE	User Equipment
UI	User Interface
UL/DL	Uplink/Downlink

UP	User Plane
UPF	User Plane Function
URLLC	Ultra–Reliable Low Latency Communication
V2X	Vehicle-to-everything
VM	Virtual Machine
VNF	Virtualised Network Function
VPN	Virtual Private Network
VPP	Vector Packet Processor
VR	Virtual Reality
WP	Work Package

3 Introduction

Deliverable D6.3 is the last deliverable of the Work Package 6 (WP6) of the Hexa-X project, being the continuation of the previous Deliverables D6.1 [HEX21-D61] and D6.2 [HEX22-D62]. It describes the work performed from M17 (May 2022) to M28 (April 2023) according to the Hexa-X project execution plan.

3.1 Objectives of the document

This document has two main objectives: (i) to report on the final evaluation of the service management and orchestration mechanisms developed in WP6 and described in Deliverable D6.2 [HEX22-D62], and (ii) to serve as a means of verification of the project Objective 3 (Connecting intelligence towards 6G) intended to deliver methodology, algorithms, and architectural requirements for an AI-native network, and in this case, focusing on the AI-driven governance. Regarding objective (i), the evaluation targets different aspects, namely: the evaluation of the M&O architectural design provided in the previous Deliverable D6.2 and an evaluation of specific KPIs, KVIs and Core Capabilities defined in [HEX21-D12] and in Deliverable D6.2, for this WP. Finally, and also as part of the evaluation, the document provides information about the main lessons learnt and some hints for future work. Regarding objective (ii), the document provides information about how WP6 has contributed to the overall Hexa-X Objectives, and specifically, to the so-called Objective 3 ("Connecting intelligence towards 6G"), in what regards those aspects related to what is our main concern in WP6, i.e., the M&O related aspects. This second document objective includes providing information about the WP6 outputs towards such Objective 3, the WP6 measurable results according to that Objective, and their related quantifiable targets.

3.2 Methodology

The above-mentioned objectives are mainly verified by implementing Demo #4 and Demo #5, which according to the Hexa-X project execution plan, are in the scope of WP6. Demo #4 main scope is *handling unexpected situations in industrial contexts*. In general, the demo targets handling unforeseen problems in a simulated industrial environment, consisting of a set of mobile robots that cooperate to complete a defined task. The unexpected situations are impairments/faults intentionally caused by the human operators in the context of the demo to test the robots' self-adaptation capabilities. This demo has been addressed from two different work packages (WP6 and WP7) that rely on the same infrastructure to address some of their specific work topics. Regarding WP6, the work done on this demo lies in applying predictive M&O mechanisms to proactively mitigate the impact of a faulty device on the routine activity of the robots, which are considered extreme-edge devices in terms of M&O. The TRL required for this demo is TRL 4 (technology validated in a lab) [TRL]. Demo #5 targets the demonstration of the *data-driven device-edge-cloud continuum management* concept. This demo focuses on a simulated road-traffic urban environment on which different scenarios have been developed, covering aspects such as the deployment of an AI/ML-driven network service to provide smart control of the traffic lights in that urban environment, application of proactive scaling polices for a URLLC (Ultra Reliable Low Latency Communications) service based on the predicted road traffic conditions, security aspects applied to this context, and the application of MLOps techniques to deploy AI/ML services. The demo focuses on using AI/ML techniques in these scenarios and also in extending the M&O scope beyond the edge, relying not only on simulations but also on practical hardware-based implementations of specific extreme-edge resources using small-scale computing devices (e.g., to simulate the traffic lights themselves and their associated controllers). The TRL (Technological Readiness Level) required for this demo is TRL 3 (experimental proof-of-concept). Both demos have been implemented and aligned with the M&O architectural design provided in the previous Deliverable, D6.2. However, from the required TRLs and the Demo's requirements in the Hexa-X project plan, a complete E2E implementation with all the architectural components and functionalities as they are described in D6.2, goes far beyond the

Demos purpose, which is with a much more limited scope. Therefore, although well aligned with the overall architecture, the demos implementation is limited to demonstrating selected topics. Demo #4 is in strong alignment with two described in D1.2 [HEX21-D12] use cases, namely: Digital Twins for manufacturing and Flexible manufacturing. Both of these use cases are related to each other. The first one describes how using Digital Twins can benefit production lines via improvements in capabilities such as management of infrastructure resources, detection of anomalous behaviour, and mitigation of critical situations. The second one focuses on allowing dynamic configuration of real-time communication services, which is essential for mobile production machinery. Demo #5 is aligned with one described in D1.2 [HEX21-D12] use case, which is 6G IoT micro-networks for smart cities. This use case focuses on the management of traffic flows in a complex local system of objects interacting with each other. Traffic Light control described in Demo #5 can be such a system. The implementation of Demo #4 and Demo #5 has been complemented with a set of lab experiments that, beyond the topics covered by the demos, are also considered interesting to explore some of the innovations introduced in Deliverable 6.2. These experiments are described in detail in Section 6. Specifically, they are three experiments concerning network energy efficiency, extreme-edge nodes discovery, and the potential usage of the Simu5G NR User Plane simulator [NSS+23] in an urban road-traffic scenario, such as the one presented in Demo #5. Table 3-1 summarizes which specific topics were addressed by each demo and the lab experiments.

Table 3-1. Topics addressed by the demos and complementary lab experiments.

Work Topic	Demo #4	Demo #5	Lab Exp.
1: Unified orchestration across the “extreme-edge, edge, core” continuum.	✓	✓	✓
2: Increased level of automation.	✓	✓	
3: Adoption of data-driven and AI/ML techniques in the M&O system.	✓	✓	
4: Unified management and orchestration across multiple domains, owned and administered by different stakeholders.		✓	
5: Adoption of the cloud-native principles in the telco-grade environment.		✓	
6: Security		✓	
7: RAN integration			✓
8: Network energy efficiency			✓

It is worth noting that most of the novel capabilities in the M&O architectural design reported in Deliverable D6.2 are addressed (items 1 to 5), with the only exception of the application of the intent-based approaches for service planning and definition (see [HEX22-D62], Section 5.3). Additionally, other topics not explicitly declared in that capabilities list nor in the Demos requirements have been included, such as the Security topic (item 6), the RAN integration (item 7), and the experiment regarding the network energy efficiency (item 8, this last one intended to evaluate one of the quantifiable targets assigned to this WP6). Based on the experience gained from the implementation of the demos and the lab experiments and on the measurements and results obtained from them, the evaluation of the main innovations raised in the previous Deliverable D6.2 has been carried out. This evaluation has focused on, validating the WP6 contribution to the overall Hexa-X objectives, the validation of the M&O architectural design, and the evaluation of selected KPIs, KVIs and Core Capabilities (from those presented in the previous D6.2) regarding the demos and the lab experiments presented here. All this evaluation information has been collected in Section 7.

3.3 Structure of the document

The document is structured as follows:

- Section 1 describes the overall scope of the document.
- Section 2 includes a list of abbreviations to support the reading process.
- Section 3 describes the main objectives of the document, the methodology followed to reach those objectives, and the document's structure (this section).
- Section 4 includes the description of Demo #4, which addresses handling unexpected situations in industrial contexts. The Demo is composed of three scenarios, which are extensively described in such section, in separated subsections.
- Section 5 includes the description of Demo #5, which concerns algorithms for data-driven device-edge-cloud continuum management. The Demo is composed of four scenarios, which are also described in that section, in separated subsections.
- Section 6 touches on complementary lab experiments concerning the benefits of the proposed proactive orchestration in the energy efficiency case, extreme-edge nodes discovery, and the integration of the RAN in Scenario 5.1.
- Section 7 is the core part of the document regarding results, targeting the final evaluation of service M&O mechanisms, mainly based on the demos and the complementary lab experiments presented in the previous sections.
- Section 8 consists of conclusions drawn on the basis of the content in the previous sections.
- At the end of the document, three Annexes describe selected mechanisms supporting demos.

4 Demo #4: Handling unexpected situations in industrial contexts

4.1 Demo overview

Demo #4, which is a cross-WP demo between WP6 and WP7, is related to both the Hexa-X Objective 3 (Connecting intelligence towards 6G) and Objective 4 (Network evolution and expansion towards 6G). The target of these objectives is to turn AI/ML into an essential component of the B5G/6G technology and to deliver enablers for an intelligent network of networks, respectively. WP6 targets specifically Objective 3, which is intended to develop AI/ML-powered enablers for orchestration and service management in order to achieve higher efficiency, increase network programmability, increase service continuity and enable new services (and revenue streams)[1]. Following this, Demo #4 has been designed to showcase a set of relevant features aligned with some of the main innovations introduced in the previous Deliverable D6.2 [HEX22-D62] and making AI/ML a core part of the M&O operations in an industrial context. With this in mind, the demonstration consists of three scenarios that will showcase the corresponding enabler:

- **Scenario 4.1**: Continuum (cloud, edge, extreme-edge) M&O of a Digital Twins use case.
- **Scenario 4.2:** Handling unexpected events using dynamic Functions Placement.
- **Scenario 4.3:** Improving service downtime and reducing costs using Predictive Orchestration.

The intention is that these three scenarios will showcase how these distinct enablers can be used together to accomplish the targets set by the related objectives. Each scenario builds upon the

1 For WP7, Objective 4 is intended to develop enablers for resource-efficient support of complex and dynamically changing availability requirements as well as Human-Machine Interaction (HMI) and fully immersive digital twins, as described in Deliverable 7.2 [HEX22-D72].

previous ones and demonstrates how the addition of each enabler helps achieve the defined objectives. In the following sections, each of these scenarios is described in more detail.

4.2 Innovations related to the demo

Demo #4 addresses three of the main innovations declared in the previous Deliverable D6.2 – Section 5.3 [HEX22-D62] for the M&O architecture, namely:

- **Unified orchestration across the "extreme-edge, edge, core" continuum.** This topic is addressed by integrating the robot infrastructure at the extreme-edge domain together with the edge and the central cloud resources. This is done using two orchestrators, one for the cloud and edge domains and another one for the extreme-edge, both of them in turn under the supervision of a higher-layer controller. The addition of the developed enablers allows the orchestration of services across the whole continuum, optimizing their placement and increasing efficiency and performance.
- **Increasing levels of automation.** This innovation is also apparent throughout this demo through the inclusion of the developed enablers. Providing novel automation mechanisms for orchestration, function redistribution, monitoring and performance diagnosis, as well as predictive orchestration, brings forth new ways to optimize the network and service deployments and utilization. The addition of these enablers paves the way for a zero-touch approach to network and service management, especially covering the distribution, placement and troubleshooting of functionalities across the various domains.
- **Adoption of data-driven and AI/ML techniques in the M&O system**. These techniques are used to generate AI/ML models and feed them with data in the context of Zero-touch automation. In turn, these models are responsible for making decisions regarding the management and orchestration of the deployed services. These tools rely on automated processes to retrieve the required data across the various domains, providing monitoring of services and components dynamically upon request.

4.3 Demo implementation

Demo #4 is designed in such a way that it can align with the different system architecture views mentioned in Deliverable D6.2 [HEX22-D62]. The various aspects of the demo are presented according to the view they align with. For reference, the software components are aligned with the proposed structural view, the functionality and algorithms are aligned with the functional view, and finally, the deployment is aligned with the physical infrastructure that has been used to implement the demo. Though most of the components are common in all three scenarios, it is also true that there exist scenario-specific components. The role of these novel components is emphasized in the corresponding component description.

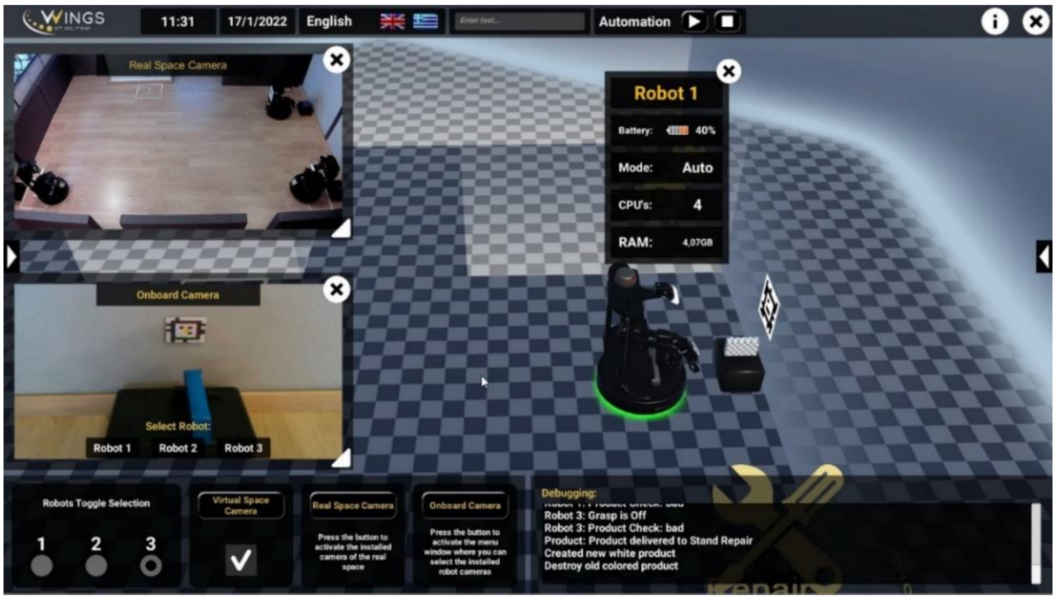
4.3.1 Scenario 4.1: Continuum (cloud, edge, extreme-edge) M&O of a Digital Twins service

This scenario is the basis on which the following two scenarios are built as well. Thus, in each consecutive scenario, all the components mentioned in the previous ones are carried over and used as well. As the title suggests, this scenario focuses on the orchestration and monitoring of the Cloud – Edge – Extreme-edge continuum.

4.3.1.1 Scenario description

In short, this demo scenario consists of a set of 3 mobile robots in a simulated industrial environment, which are able to work as Digital Twins, meaning that they can "copy" and execute the actions and movements required by humans through a remote Human-Machine Interface (HMI), thus avoiding the human presence in the industrial environment itself, which could be inconvenient or even dangerous in some cases. To make this possible, this HMI has been implemented through an advanced Virtual Reality (VR) User Interface (UI) that can be used to visualize the whole industrial environment in real-time and with 3D graphics, as well as the robots themselves, providing a great level of detail (see Figure 4-1). This way, the end-user can manage the robots and also detect issues and fix them through teleoperation.

Besides the use-case itself, this demo scenario also aims to demonstrate one of the main innovative concepts introduced in the previous Deliverable D6.2: the continuum M&O concept through the Cloud–Edge–Extreme-edge continuum, being the robots the main elements of the extreme-edge infrastructure itself. This contrasts with the common state-of-the-art approach, which typically addresses each domain in a separate way.



Robot 3: Grasp is Off
Robot 3: Product Check: bad
Product: Product delivered to Stand Repair
Created new white product
Destroy old colored product

Figure 4-1. Digital Twin interface.

The information regarding the status of the robots, battery capacity, CPU usage, RAM usage and storage capacity is sent to the Digital Twin interface and illustrated in an easy-to-view way. The UI can further display the number, names, health status of the robotic services running at each robot in the industrial environment and even display alerts when an impairment is identified. Additionally, for the creation and control of the industrial Virtual Reality (VR) environment and the Digital Twins, the Unity game engine [UNI] is used. The Automatic Guided Vehicles (AGVs) are equipped with 5G NR modems that are used to communicate with the cloud, housing the industrial automation service and the edge, where the AGV controllers are located.

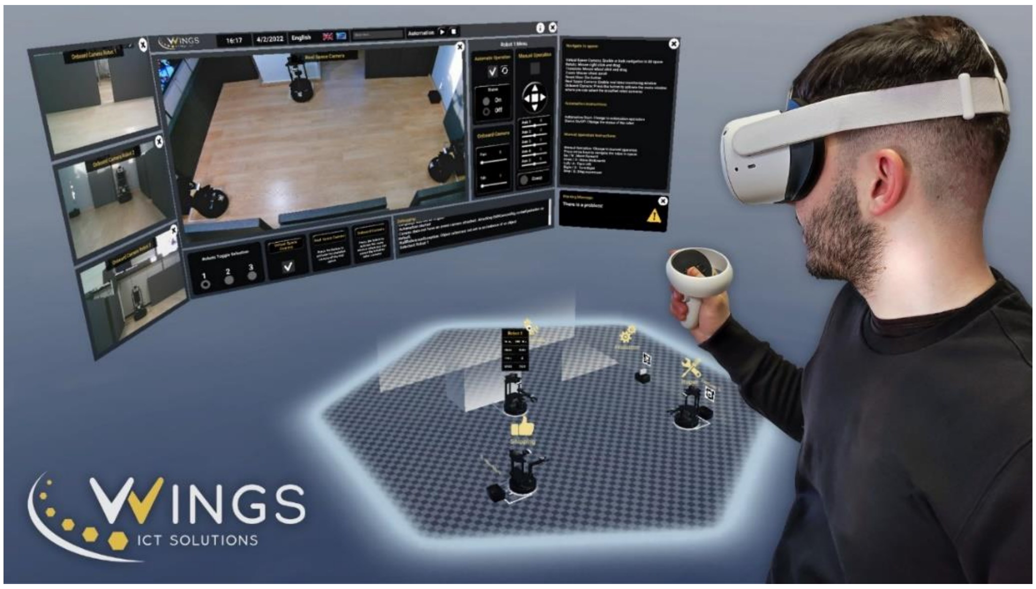


Figure 4-2. VR control of Digital Twins.

4.3.1.2 Software components

The software components used for the scenario implementation have been designed as standalone and loosely coupled modules to allow for their independent development, deployment and scaling when necessary. Since the three scenarios use, more or less, the same components, the software architecture is described once here and it will be referenced in the following scenarios. An integrated view of the software architecture and the various component interactions is pictured in Figure 4-3.

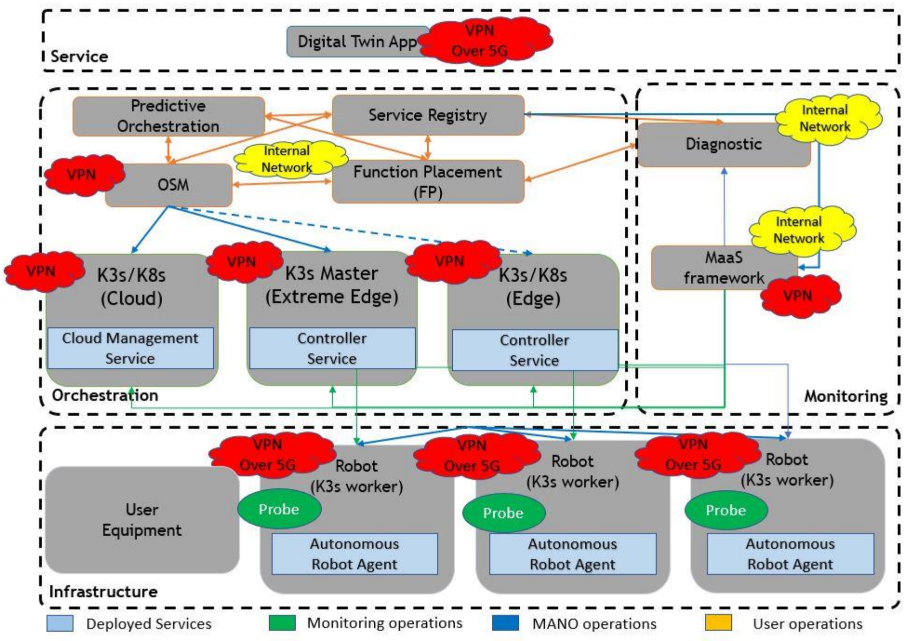


Figure 4-3. Demo #4 software architecture.

For the implementation of the scenarios, and specifically for the unified orchestration of the Cloud – Edge – Extreme-edge continuum, the Open-Source MANO (OSM) [OSM] was selected as the baseline orchestration stack. On top of OSM, a few additional components were developed in order to handle the higher-layer intelligence operations, such as differentiation of the various domains, deployment on the individual domains, management operations, etc. In addition, components such as Intelligent Orchestration and Diagnostics support the orchestration procedures for automatic deployment, scaling and migration of services when needed. Intelligent orchestration is a superset of other components like the AI algorithms (i.e., Function Placement) and orchestration manager, where all interfaces with other components and tools are located for enabling the various actions and the service registry. These will be explained in detail later in this document. At the lowest layer, the Virtualized Infrastructure Managers, in this case, OpenStack [OST] and Kubernetes (K8S) [KUBa], work together with OSM deploying the Virtual Network Functions (VNF) or Containerised Network Functions (CNF). The deployed functions for this demo include the components of the industrial automation service. The components comprising that service are the Industrial automation cloud component, the edge-level controller and the autonomous robot agents. These functions provide the external management of the automation service, the edge-level group coordinator and the individual robot agents, respectively. Additionally, the MaaS component collects all system, network and robot metrics for analysis by the Diagnostic component. K8s is deployed as a cluster, where extreme-edge devices, edge and cloud are introduced and managed by the system. For the extreme-edge devices, a lightweight yet powerful certified K8s distribution designed for production workloads, called K3s [K3S], was selected. Monitoring probes are also used to monitor the infrastructure, network and services that are running inside the system. The probe is the ultimate tool to capture and analyse data in real-time to help system operators find the sources of any slow-downs or performance bottlenecks before they begin to affect the system. Collected data can be processed locally and published to the target output, in our case, the Monitoring system, which is the central point of all logs, metrics, and other types of data from the system. The probe is the ultimate tool to capture and analyse data in real-time to help system operators find the sources of any slow-downs or performance bottlenecks before they begin to affect the system. Collected data can be processed locally and published to the target output, in our case, the Monitoring system, which is the central point of all logs, metrics, and other types of data from the system. Different actions, events and alerts can be generated by the system based on the analysis done.

The group containing the Service Registry, Predictive orchestration, Function placement, and Diagnostic components for the purposes of this demo is referred to as Intelligent Orchestration. This fabric of functionalities provides added value on top of the M&O capabilities. It is the higher-layer controller logic that is constantly fed data from the MaaS framework and propagates its decisions for potential actions to the orchestrator (OSM) for enforcement on the infrastructure. In the Infrastructure layer, there are three robots that are being used as workers for the industrial automation context. These robots receive and execute tasks based on the requirements of the production line inside this context. The User Equipment are mobile terminals, in this case, a laptop and a set of VR glasses with controllers, used by local or remote human workers or technicians to control, if necessary, overview or provide a technical examination of the robots. Finally, the Digital Twin application is a desktop and VR application that interfaces with robots as part of the industrial automation service and provides the tools to humans to perform the aforementioned actions. Detailed descriptions of each component used in this scenario can be found in the following subsections. Some of the components, namely the Predictive Orchestration, Function Placement and Diagnostic components, is described later in the next scenarios, where they are introduced respectively.

Digital Twin App

The Digital Twin application has been developed using Unity [UWS], a 3D engine for creating real-time 3D games, apps, simulators, and it is even used in films, automotive, architecture, and more. The interface fully depicts the industrial environment with 3D graphics providing remote monitoring and control of the environment by displaying in real-time and in great detail industrial systems, robots, their parts, movements, forces, interactions, and all other assets. A human can be involved (HMI – Human Machine Interaction) in these industrial tasks with the use of a Digital Twin App via Virtual Reality (VR) technology with immersive, realistic 3D graphics (Figure 4-4). Through the HMI the user can receive notifications, observe and interact with their digital twins, examine their exact location on the site and their status condition (battery level, mode, RAM, CPU, running services, etc.) at all times. Video is streamed in real-time from the cameras of the robots as well as from other cameras placed inside the industrial environment. By using Virtual Reality (VR), the remote interaction with the factory becomes more interactive. The user, with the help of special glasses, can "touch" the robots virtually, come close to them, control the different parts of the robot, supervise, and adjust its parameters. Finally, using the remote control, a user can carry out a task “manually”, solve problems, or in case of faults, move a robot to a specific area (e.g., a repair point) where a mechanic can fix the problem. More details about the Digital Twin and the VR application are described in Deliverable D7.3 (section 6.1.7), which is planned to be released alongside with this document.

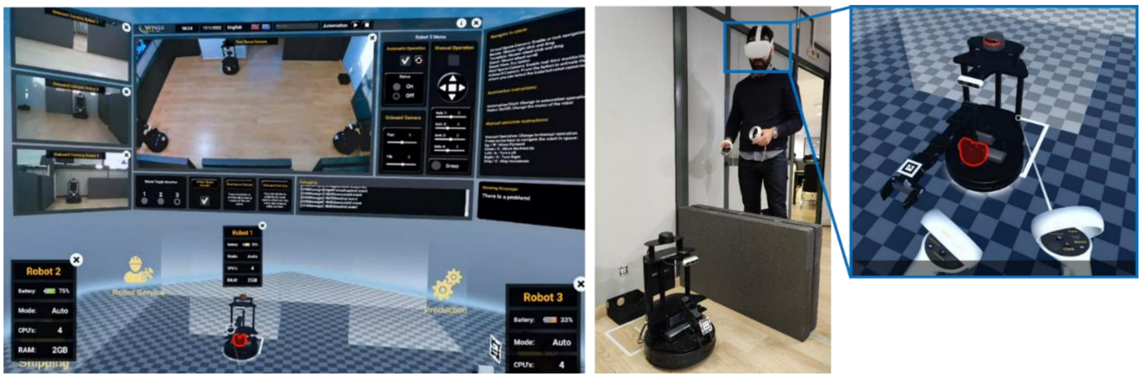


Figure 4-4 Digital Twin Application Interface

Deployed Services

The main target of the M&O operations are the deployed service components that comprise the industrial automation service used in this demo. These components are (see Figure 4-3):

- the cloud management service;
- the controller services;
- the autonomous robot agent services.

Based on their functionalities, each of these components can be deployed on the corresponding infrastructure. The cloud management service is resource intensive, as it hosts most of the AI functionalities of the industrial automation service. So it's better suited to be hosted on the Cloud. It is also the main gateway for the users to get access to the provided functionalities, such as teleoperation of the robots, remote inspection, and monitoring. The controller service implements some communication and coordination functionalities between collocated and collaborating robots and can be deployed both on the Cloud and on the Edge. Finally, the autonomous robot agents, as the name suggests, are deployed on the individual robots acting as the proxy between the management services and the robot worker. These agents also allow the deployment of more specific services that can implement various worker roles in the industrial context. These services are practical implementations of robotic actions written in python and C++ that work on top of ROS (Robotic Operating System) [ROS] designed and developed in a microservices approach. Services can use sensors and servos to detect and grab objects, map the environment, navigate, collaborate with other robots, etc., inside the industrial environment. The conversion of roles to services allows the extension of the M&O reachability to intrinsic service components, enhancing the programmability of the extreme-edge.

MaaS framework

In terms of architecture, Figure 4-5 shows the design of the MaaS framework, which is composed of three main parts: the MaaS Client(s), which implements the front-end; a MaaS Server; and the Bridge, which translates the MaaS server probes' requests to the targeted cloud platform technology.

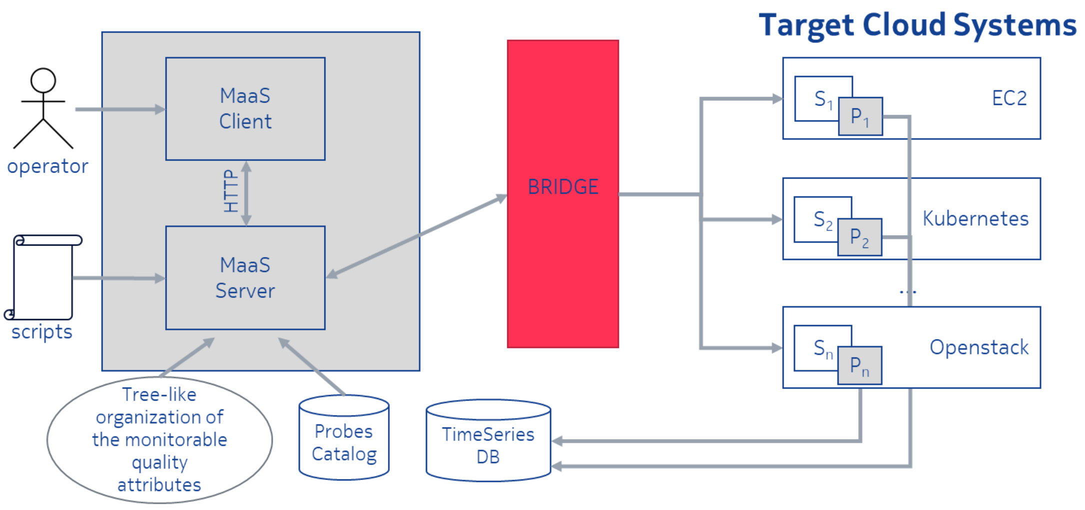


Figure 4-5. Main components of the MaaS architecture.

In this scenario, the cloud platforms are the K3s cluster deployed on each robot. As a whole, the MaaS Platform works like this: the MaaS Client selects its monitoring goals. This selection is received by the MaaS server and transformed into a technology-agnostic probe deployment from the probes available on the probes catalogue. The MaaS Server supports multiple monitoring strategies. Either the probes are deployed as a sidecar or already provisioned in the running VM (Virtual Machine) or container. In most cases, the probe is deployed as a sidecar since sidecar containers can share resources with the target containers that are needed to be monitored, and monitoring can be performed seamlessly. The MaaS server selects the probes from the probes catalogue and the way the probe is deployed. The nature of the sidecar, VM or container is let to the bridge. The probes are stored in a probe catalogue and associated with metadata that allows the MaaS server to select the deployment decisions. The metadata information contains the KPIs that the probe can collect and its deployment option (as a sidecar or not). The nature of the sidecar, VM or container is let to the bridge. The Bridge is the component responsible for mapping the pattern identified by the MaaS Server into a number of operations to deploy the necessary set of probes. In particular, the Bridge implements CRUD (Create Read Update Delete) operations on the set of probes present in the targeted cloud platform. The deployed probes push data into a Time Series Database (TSDB) that stores the collected data. The MaaS can integrate any TSDB technology as long as probes are properly configured. The MaaS also supports the possibility of using a pub-sub channel to decouple the probes from the TSDB.

Service Registry component

The Service Registry component is developed with the purpose of providing a well-defined directory of available/running services so that the system always has the latest information on specific values and requirements of these services. Additionally, it acts as a common data storage among all the components for synchronizing operations between them, management, and status monitoring. All the services that get registered to this component become visible to the other components as well. For each registered service, the following information is provided:

- descriptors to be passed to the orchestrator for the actual deployment;
- metrics/KPIs of interest to be monitored;
- resource requirements for the deployment of the service;
- capabilities requirements for the deployment of the service;
- information regarding dependency on other services;
- exposed endpoints for discoverability and programmability;
- performance constraints for the diagnosis;
- criticality;
- resource/power usage profiles.

After the registration, each component can retrieve the corresponding required information from the Service Registry in order to perform its tasks.

Service Repository component

The Service Repository implements the role of the common repository for all the components and service artefacts required in order to deploy the registered services like the Docker images, helm charts, etc. This repository acts similarly to a localized Content Delivery Network (CDN) for all the infrastructure across the Cloud – Edge – Extreme-edge continuum. Utilizing this component, the functionalities under the common M&O continuum are able to be distributed and placed on any of the managed resources, given that capabilities and hardware constraints are taken into consideration as well.

4.3.1.3 Functional behaviour

All the developed components have distinct functionalities that complement each other and are focused on the intelligent and automated management & orchestration of infrastructure and services.

As described, artificial impairments (high CPU/memory/disk load, high latency, low battery, hardware stress, etc.) are introduced to the system. For the software to be able to identify these impairments, custom probes capable of monitoring all the required metrics are deployed as well. These probes are developed with functionalities to monitor all the required metrics. More information on these probes is found in the MaaS description below. To support the defined scenarios, groups of operations have been created to showcase the various stages of the M&O functionalities. These groups are:

- service registration & instantiation;
- functions placement operations;
- predictive orchestration operations.

Initially, the Service registration & instantiation stage includes all the operations required for the deployment of the industrial automation service. The sequence diagram for this stage can be seen in Figure 4-6. After the instantiation of the service is finished, the service is added to the list of services under monitoring. This is a requirement for the second stage and part of Subsection 4.3.2, i.e., the Function Placement operations. The functionality of each component from those described in the previous Section 4.3.1.2 is described in the following subsections.

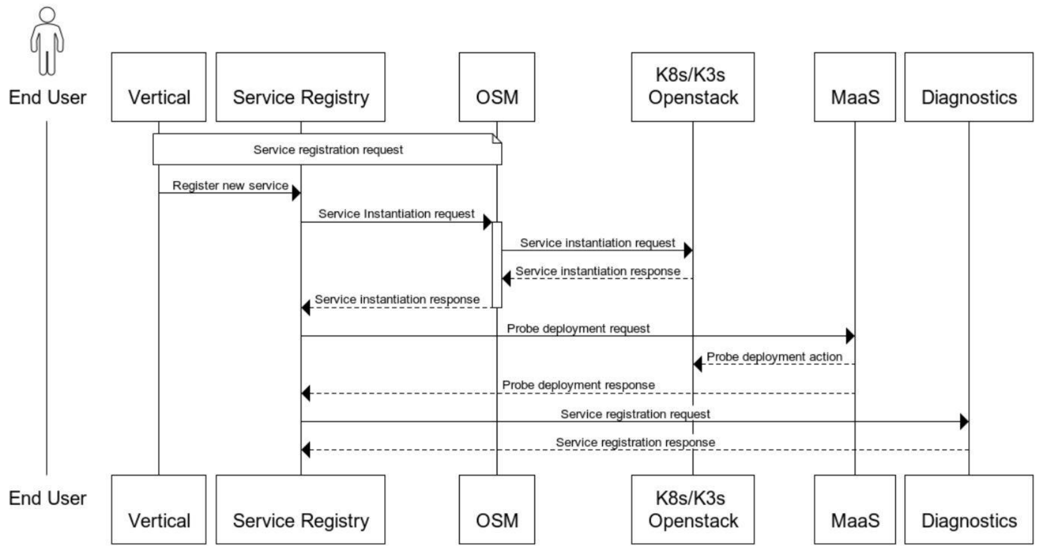
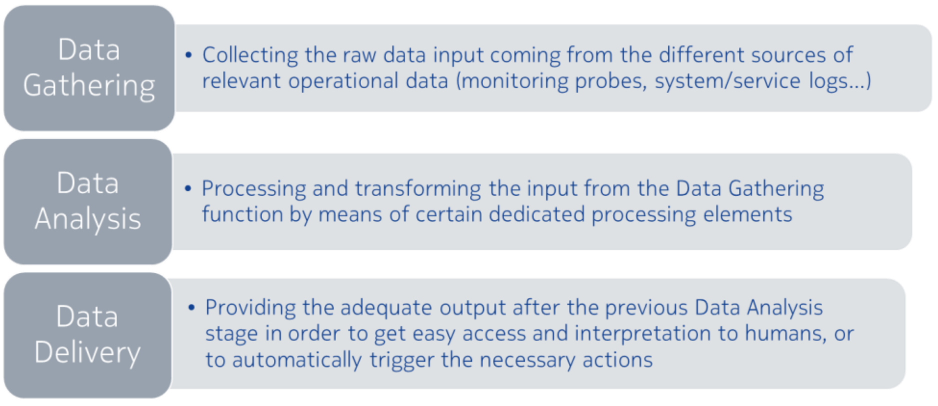

End User
Vertical
Service Registry
OSM
K8s/K3s Openstack
MaaS
Diagnostics
Service registration request
Register new service
Service Instantiation request
Service instantiation request
Service instantiation response
Service instantiation response
Probe deployment request
Probe deployment action
Probe deployment response
Service registration request
Service registration response
End User
Vertical
Service Registry
OSM
K8s/K3s Openstack
MaaS
Diagnostics

Figure 4-6. Service registration & instantiation sequence diagram.

<u>MaaS framework</u>

The MaaS framework offers three main functionalities summarized in Figure 4-7.



- Collecting the raw data input coming from the different sources of relevant operational data (monitoring probes, system/service logs...)
- Processing and transforming the input from the Data Gathering function by means of certain dedicated processing elements
- Providing the adequate output after the previous Data Analysis stage in order to get easy access and interpretation to humans, or to automatically trigger the necessary actions

Figure 4-7. Main functionalities offered by the MaaS framework.

For the Data Gathering, the goal of the MaaS platform offers the possibility to manage the monitoring goals in a declarative manner and independent from the underlying cloud platforms (OpenStack or K8ss) and technologies. The MaaS provides a unique entry point for the different stakeholders (platform owner, service owner etc.) to monitor different KPIs. The MaaS offers the following features:

- **Technology agnostic**: the MaaS is a general framework that is designed to support multiple target cloud platforms and multiple probing technologies. One can cite Prometheus Exporters [PAE], Elastic Stack Beats [Ela19], or a custom probe like the one deployed in this Demo #4. This probe is capable of collecting all the required metrics to support the scenarios in the industrial context. In this context, the selected metrics that are used as input for the intelligent orchestration mechanisms, along with the service descriptors, capabilities and constraints, are:
    - resource metrics (CPU, memory, disk);
    - network metrics (latency, packet loss, UL/DL data rate);
    - power consumption;
    - service KPIs:
        - industrial operation cycle time;

  - packages handled per min/hour;
  - incidents per min/hour;
  - availability.

  The deployment of the probes is handled by the automated mechanisms provided by the MaaS framework.

- **Model-driven**: the monitoring goals are specified following a tree-like model derived from the ISO 25011 service quality standard [ISO2011, ISO2017]. The model defines multiple quality attributes that are decomposed into finer-grained concepts until reaching measurable properties, as it is depicted in Figure 4-8. Indicators can be selected at any level of the tree. The selection is mapped automatically into the measurable properties associated with the leaves of the selected subtree. This request is then transformed into a set of probes that are deployed in the target platform or system.

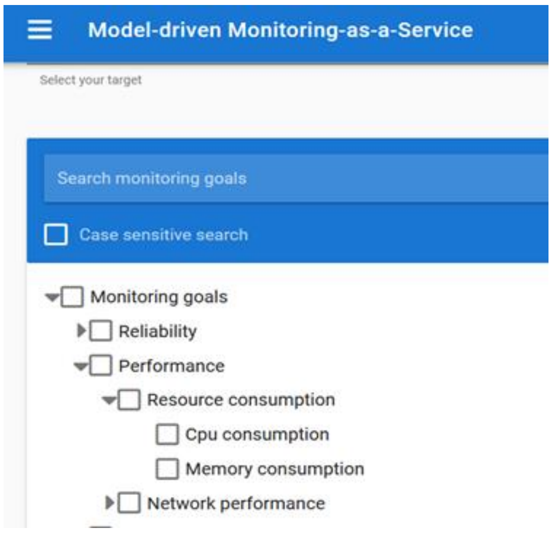


Figure 4-8. Tree-like organization of the monitorable quality attributes.

```
- name: Hexa-X
  monitoring_subgoals:
    - name: infrastructure
      cloud_properties:
        - cpu_utilization
        - memory_utilization
        - disk_utilization
        - latency            -
```

Figure 4-9. Example of MaaS monitoring goals.

Figure 4-9 introduces an example of monitoring goals. Of course, the MaaS supports many other goals, and use-specific goals can also be added.

Service Registry component

The main functionality of this component is to store and share information among the components of the system. By using the service registry, the problem of two components having different values, identifiers or even semantics has been avoided by providing a central reference place for the information it needs, enabling thus greater continuity of the intelligent orchestration system. Its main functionalities are:

- Registration, where data, metrics, configurations, requirements, etc., are kept and can be accessed by other mechanisms of functions.
- Retrieval, where any component can retrieve the service-specific information it requires.
- Modification, where the contents of the services that are stored to match the latest versions and updates can be updated. For example, every time a service is moved from one node to another, the service registry needs to have the latest changes available for all components to reflect the latest deployment status.

- Filter or query, where the output is processed to match a specific metric or value.
- Deletion is the process of removing irrelevant data from the service registry. For example, if a service is not used, the associated records containing all the information for orchestration, monitoring, and other components can be removed.

Specifically, after a new service, in this case, the industrial automation service, is registered, OSM is triggered to deploy the new service on the selected infrastructure. After deployment, the service components perform their internal operations for discovery and coordination and then extreme-edge infrastructure, the robots, become available for operation through the Digital Twin application.

Service Repository component

The functionality provided by this component is that of a repository for all the required software and artefacts for the deployment of services. Each service that is registered in the Service Registry requires sources for the artefacts to deploy. These sources can be local or remote. Having this local repository with all the required artefacts enables immediate deployment of new or existing services, especially on "fresh" nodes that have not been used before.

Specifically, this demo implements the role of a Docker images repository hosting the container images for the various services utilized in the industrial automation context. Furthermore, it also performs the role of the Helm chart [HELM] repository for the various K8s charms utilized for the deployment of the same services. Finally, it can also be used as storage for the descriptors, configurations and package/binary files that might be used during the scenario workflows.

4.3.2 Scenario 4.2: Handling unexpected events using Functions Placement

This scenario introduces novel mechanisms to detect and handle unexpected events during the operation of the industrial automation service that has been selected for this use case.

4.3.2.1 Scenario description

This scenario aims to demonstrate how AI/ML enablers, for anomaly detection and performance degradation analysis, along with increased automation and programmability, can be utilized to further increase the efficiency of network and/or service operations, in this case, industrial operations, using closed-loop control mechanisms. These mechanisms rely on monitoring and performance diagnosis of the various services and components running on the infrastructure and are responsible for reconfiguring and redeploying services and functionalities in order to optimize their performance and achieve the targeted KPIs/KVIs.

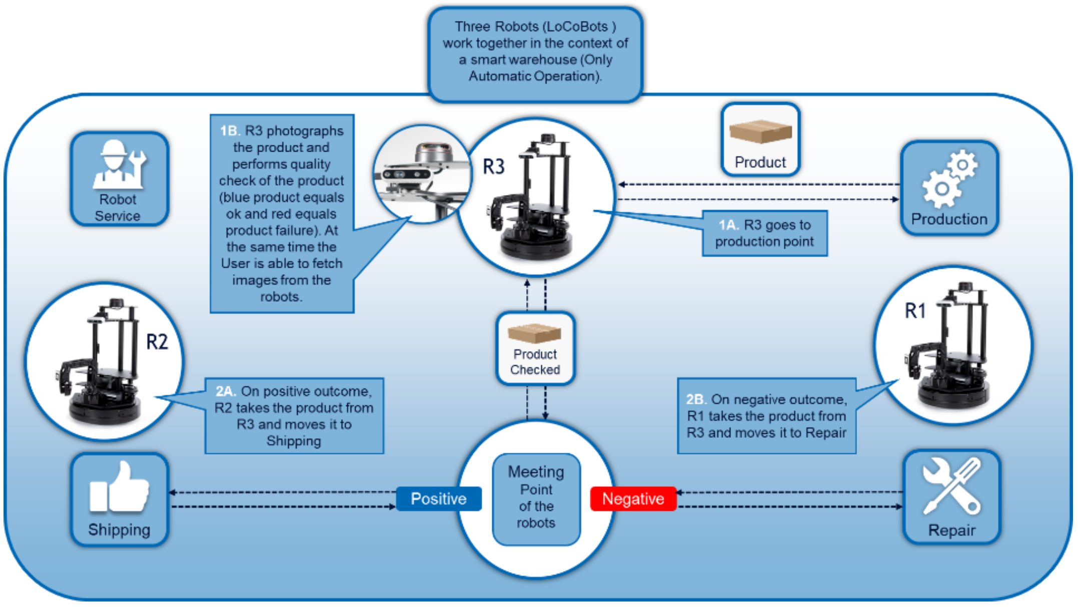
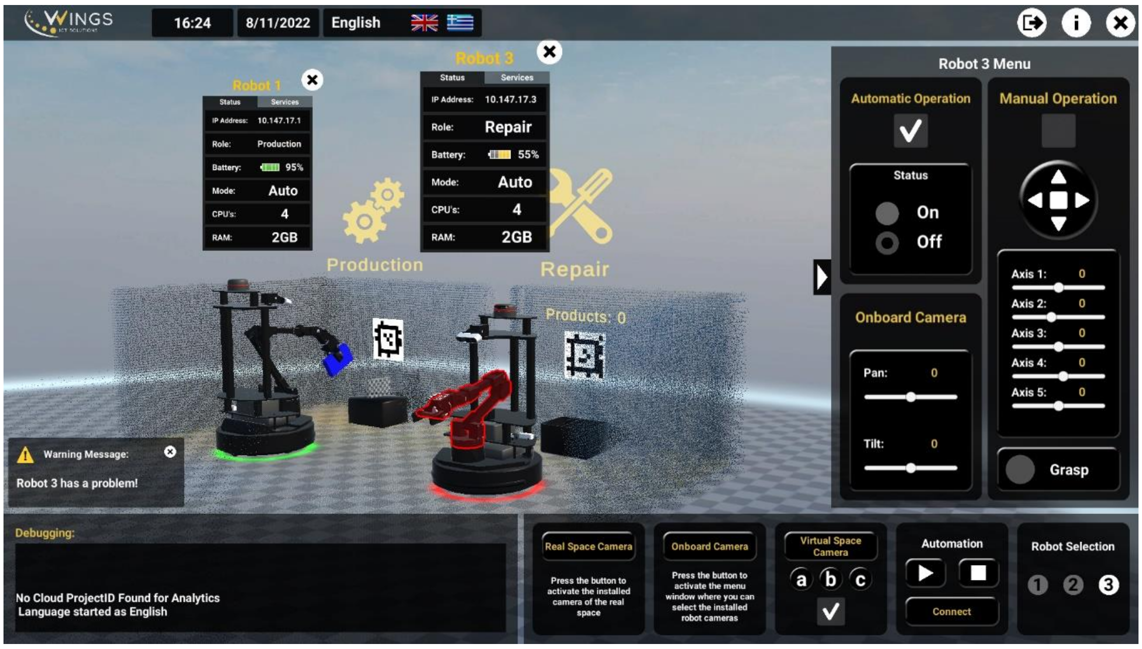

Three Robots (LoCoBots) work together in the context of a smart warehouse (Only Automatic Operation).
Robot Service
1B. R3 photographs the product and performs quality check of the product (blue product equals ok and red equals product failure). At the same time the User is able to fetch images from the robots.
R3
Product
Production
1A. R3 goes to production point
R2
Product Checked
R1
2A. On positive outcome, R2 takes the product from R3 and moves it to Shipping
2B. On negative outcome, R1 takes the product from R3 and moves it to Repair
Shipping
Positive
Meeting Point of the robots
Negative
Repair

Figure 4-10. Robotic scenario

Additionally, these mechanisms are used to show how unexpected situations can be handled in an automated way without requiring human intervention while still allowing the option for a "human-in-the-loop" workflow. Beyond the implementation itself, what this scenario demonstrates is the benefits of deploying closed-loop control mechanisms that rely on real-time data from the infrastructure and deployed services, as well as AI/ML mechanisms, in order to optimize the deployment and performance of the same services. Doing so, the accomplishment of target KPIs/KVIs can be monitored, and corrective actions can be taken immediately, or even pre-emptively, upon detection/prediction of irregularities instead of relying on delayed human intervention. To showcase the various functionalities in the context of an industrial environment, in this scenario, impairments to the operations are injected artificially. These impairments can be applied instantly or gradually so as to better showcase the triggered responses from the corresponding handling component, the diagnostic and predictive orchestration components in this case. These impairments include, but are not limited to, the following:

- resources stress (CPU/memory/disk);
- network stress (artificial latency/packet loss);
- artificial low battery;
- artificial motor stress.

For the demo implementation, an emulated industrial production line has been built using three mobile robots, with three locations assigned as goals for their respective roles of the robots and placeholder objects that are to be transferred between the different target locations: Production (quality checking), Shipping, and Repairing, based on the role that each robot implements.



Robot 1

IP Address:	10.147.17.1
Role:	Production
Battery:	95%
Mode:	Auto
CPU's:	4
RAM:	2GB

Robot 3

IP Address:	10.147.17.3
Role:	Repair
Battery:	55%
Mode:	Auto
CPU's:	4
RAM:	2GB

Warning Message:
Robot 3 has a problem!

No Cloud ProjectID Found for Analytics
Language started as English

Press the button to activate the installed camera of the real space

Press the button to activate the menu window where you can select the installed robot cameras

Figure 4-11. An anomaly detected during robot operation.

Figure 4-10 depicts the described robotic scenario. Further information regarding the industrial environment and the robotic operations can be found in Deliverable D7.2 (to be released together with this Deliverable D6.3) in section 4.2.2. The mobile robots are also equipped with compute resources allowing the M&O systems to deploy and manage services on them as extreme-edge nodes. There is also a UI that can be used to provide information regarding technical malfunctions or performance degradations to technical personnel in case a remote intervention or maintenance is needed, like in the case of the malfunction shown in Figure 4-11. In this example, one of the robotic arms is experiencing increased stress caused by external human action for demonstration purposes, which is detected using the collected stress metrics from the arm's motors. All the software components in these domains can be managed and orchestrated by the orchestrator.

4.3.2.2 Software components

In this scenario, the previously mentioned components from Scenario 4.1, MaaS, Service Registry, and Service Repository, are also used. The newly introduced components are the Diagnostic and the Function placement components, shown in Figure 4-3. For details on those components, refer to Subsection 4.3.1.2. The details for the new components introduced in this scenario can be found in the following subsections.

Functions Placement component

The Function Placement component (which is described in detail in Deliverable D7.3, sections 6.1.4 and 6.1.5) is developed with the purpose of optimizing the placement of services and their components across the available infrastructure, either at the cloud, edge or the extreme-edge domains, as seen in Figure 4-3. In the industrial context of this demo, for example, in the case of an unexpected situation, like the case where a task/functionality is in pending mode or is executed slowly, it is crucial to reallocate this malfunctioning task to one of the other nodes/robots to avoid possible downtimes. Accordingly, in the case where a robotic device goes out of order, there should be a component/algorithm responsible for redistributing the functions to the rest of the nodes/robots/units in an orchestrated manner and with minimum cost. For the correct operation of this component, interfaces with the Diagnostic, Monitoring, and Orchestrator components are needed. These interfaces are implemented as typical REST interfaces between the corresponding components. The interface with the Diagnostic component provides the trigger input for when services or components need to be moved in order to alleviate undesired situations or performance degradations. The interface with the Monitoring component, the MaaS framework, is required so that the Function Placement component can retrieve the current status of the infrastructure available and services in operation. Finally, the interface with the Orchestrator is required for the enforcement of the decision for the optimal placement of one or more services or components.

Diagnostic component

The Diagnostic component’s purpose is to carry out a performance diagnosis of a deployed service without extensive knowledge of the service’s functionality, metrics, and overall usage. Thus, it can comprise a useful tool that can provide valuable information to the components responsible for the optimal placement of the services and components. It has been developed specifically for monitoring a service/node and is characterized by a set of metrics/KPIs in order to detect anomalies in the observed behaviour or performance degradations. To accomplish that, this component utilizes information retrieved from the monitoring system, the MaaS framework. After the detection of such an event, a trigger is generated for the other components, specifically the Function Placement (FP) component, for an evaluation of the optimal placement of the service and its components. The optimization goal of the placement can be configured for a service based on predefined policies like minimizing the number of robots used, optimizing power consumption on the robots and optimizing based on the robot location. Additionally, if the services are not tightly coupled with the robots and can be offloaded on the Edge or Cloud compute nodes, then additional options exist regarding minimizing the latency of the service, maximizing the number of industrial tasks completed and so forth. To accomplish the detection of these events, the Diagnostic component utilizes different AI algorithms that have been implemented as interchangeable submodules to be used in conjunction with this component, also allowing the extension of the list of supported algorithms. This allows the extension of this mechanism with additional algorithms following common interface and data information models. The currently implemented algorithms include Self-Organizing Maps (SOMs) [SOM] and Depth-based spatial clustering of applications with noise (DBSCAN) [DBS] as an alternative for the clustering of the observed data, as well as a custom algorithm for topological investigation, correlation and root cause analysis based on adjacency lists. The algorithms are described in more detail in subsection 4.3.2.3. Additionally, if the necessary requirements are met, a root cause analysis process can also be executed in order the detect the origin of the observed anomaly or degradation more accurately. The combination of these algorithms is capable of providing additional analysis capabilities by exploiting the advantages of the SOM tool (unsupervised learning, heterogeneous input) and utilizing additional information.

4.3.2.3 Functional behaviour

This scenario relies on the already provided functionalities from Scenario 4.1 and introduces new workflows on top of them. During this stage, the running services, along with the available infrastructure, are continuously monitored for changes, errors, anomalies, or performance degradations. When such a case is identified, the Function Placement mechanism is triggered in order to compute the optimal placement for the service or services in question. The sequence diagram for this workflow is shown in Figure 4-12.

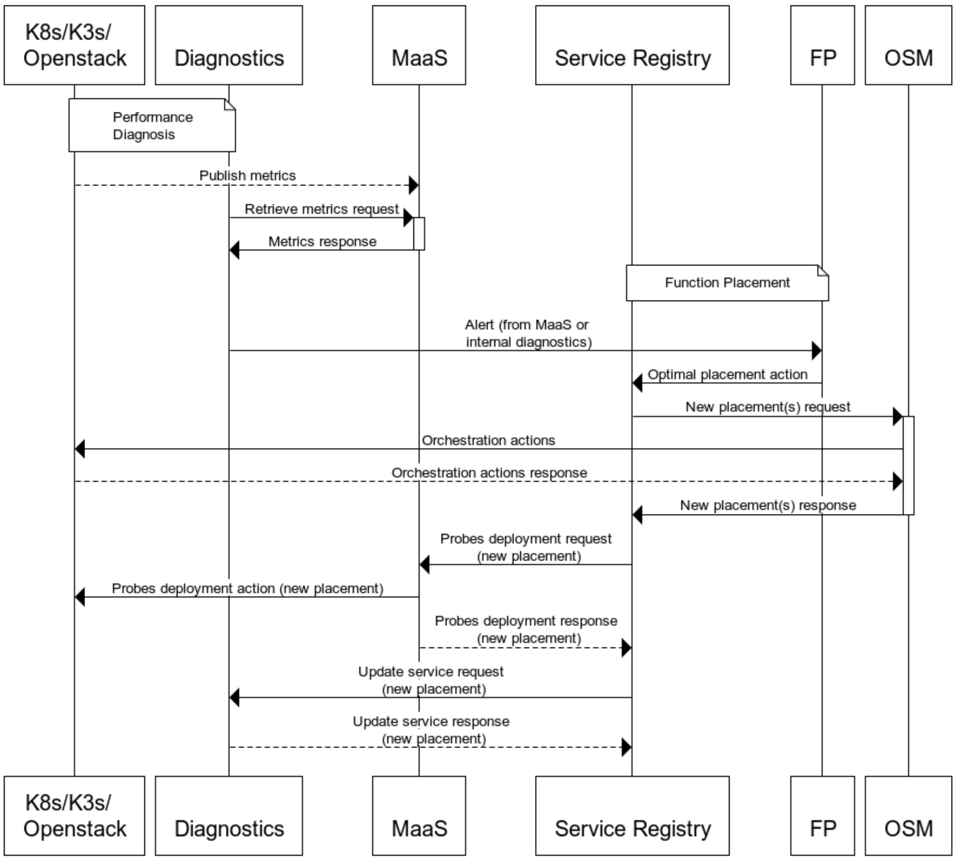

K8s/K3s/ Openstack
Diagnostics
MaaS
Service Registry
FP
OSM
Performance Diagnosis
Publish metrics
Retrieve metrics request
Metrics response
Function Placement
Alert (from MaaS or internal diagnostics)
Optimal placement action
New placement(s) request
Orchestration actions
Orchestration actions response
New placement(s) response
Probes deployment request (new placement)
Probes deployment action (new placement)
Probes deployment response (new placement)
Update service request (new placement)
Update service response (new placement)

Figure 4-12. Function Placement workflow

A detailed description of the new components' functional behaviour that performs these new functionalities can be found in the following subsections.

Diagnostic component

The Diagnostic component is utilized as a part of the intelligent orchestration group of functionalities. After the services and components of the industrial operation scenario have been registered and deployed on the corresponding domains, the Diagnostic component is triggered, by the Service Registry, in order to start performing analysis on the data generated by the service. During the industrial automation service operation defined in the scenario, multiple metrics are generated from the utilized nodes, for instance, CPU/RAM/disk/network utilization, as well as higher-level KPIs that have been defined for the service, for example, finished work cycles, number of examines items, number of malfunctions, etc. These metrics and KPIs are collected and exposed to the M&O system from the probes deployed using the MaaS platform. While the infrastructure and network ones are retrieved independently from the respective resources, for the service-specific KPIs, a common way of exposure and retrieval needs to be set for both the infrastructure provider and the service provider. For this demo, the high-level KPIs and the endpoints they are exposed to are declared in the initial definition of the services registered in the Service registration stage. This ingested information is then passed through a vectorizer, which is a module that vectorizes this information in order to be spatiotemporally correlated. This means that the metrics and KPIs of a specific time slot are grouped together to provide a snapshot of the system and the service for that specific time slot, a vector.

The created vectors contain one group of information per node as well as some general information of the vector. Each node is represented in the vector by its metrics and some general information about the node, including the anomaly detection analysis. For example, a vector can contain the groups: *vectorID* (a unique ID for each node), *Timestamp* (the timestamp of the snapshot the vector represents), *Service* (The name or ID of the service) and *vector_data*, the structure that contains the metrics of each node. The *vector_data* structure contains a *Nodes* list, where each element is a set of data for each node, and each set contains the values shown below:

- *nam*e: name of the node;
- *state:* node's state according to the outlier detection decision;
- *outlier:* boolean value to show if a node is detected as an outlier or not;
- *distance:* distance between the node's metrics and the respective neuron's weights;
- *threshold:* the predetermined threshold for the node's distance, used for outlier detection;
- *percentage:* the deviation between each metric and the respective neuron weight;
- *metrics:* list of node metrics, where each element contains information for the *name, value, unit* and *timestamp* of each metric.

Since each vector represents the service's state at a given moment in time, the number of vectors propagated to the models is determined by the metrics collection frequency. An unsupervised learning process is taking place (as described in [SOM2]) when running a service for the first time, assuming the service runs in a normal environment, i.e., using non-anomalous data to train the models. The training is executed in real-time, starting from the instantiation of the service until a specific point is reached (determined from testing in an equivalent infrastructure/environment), where it is assumed that the learning process is completed. From that point on, the model can be used to detect anomalous sets of metrics of a node inside a given vector. Topologically, the training and inference stages benefit from increased resources, and such takes place at the resource that the Diagnostic component is deployed in the Cloud. The generated 2D map of neurons, created after the feature set has been created from the ingested metrics, is used during the inference stage in order to detect anomalous states in the ingested vectors and trigger the root cause analysis process. As the selected algorithm for the clustering functionality, the SOM utilizes the concept of an artificial neuron map, where each neuron is characterized during training by its individual weights., i.e., the values that correspond to each input metric and are determined during training. The procedure to be followed in order to identify abnormal behaviours of the nodes is described by the following steps:

- Step 1 - Data collection: The input data, which are the service's metrics/KPIs, are collected and correlated spatiotemporally as vectors.
- Step 2 - Initialization of the topological map: The topological map is composed of 25 neurons (assuming that the variety of metrics/KPIs is under a specific range). Each neuron is initialised by a random set of weights.
- Step 3 - Competition learning process: For each input vector, the Euclidean distance between the weights of the neurons is computed. The neuron whose weights are most similar to the input vector is selected as BMU (Best Matching Unit). This step is repeated until all input vectors are examined and assigned to their BMUs.
- Step 4 - Cooperation learning process: Through this step, all the sets of weights are updated. Steps (3) and (4) are repeated until several iterations have passed.
- Step 5 - Detect faults: After training, the quantization error for each input vector is calculated. The quantization error is the distance between the input vector and the BMU's weights. A self-adjusted threshold (the parameters for self-adjusting are set during testing) is used to determine if this distance is long enough for the input vector to be characterized as an outlier. The nodes state is diagnosed using this comparison.

The SOM algorithm was chosen, as it is able to combine and analyse various performance metrics (such as CPU utilization, memory utilization, disk I/O, network interface I/O, request latency etc.) to learn their underlying patterns and ultimately make decisions about the health status of VNFs

(Virtualized Network Functions) and CNFs (Containerized Network Functions) instances of services without prior knowledge of specific thresholds. The examined services, in this case, are the services that comprise the industrial automation service, namely the cloud management service, the edge controller service and the autonomous agent services. Upon detection of such events, and if the required information has been provided during service registration (e.g., available topology information and information for the internal service elements), a topological investigation is launched to try to identify the possible causes for those events. The selected algorithm for topological investigation is based on Root Cause Analysis (RCA) [RCA]. The basic principle of the algorithm is to check the reachability between the nodes (for example, instances that belong to VNFs), using each node's health status that is determined by the fault detection performed in the SOM module and the network topology. The RCA module can also enable fault localization by identifying connected nodes that affect each other's status, e.g., the non-healthy node connected to a healthy node and both being detected as non-healthy. The selected algorithm uses the health status provided to label the service nodes as "Healthy" or "Unhealthy". An extra label, "Unknown", is applied for the "Unhealthy" nodes that may be affected by other respective nodes. This is performed using an adjacency list that is obtained from the network topology. Specifically, an n-node undirected graph represented as an adjacency list is created using the virtual links of the topology. The nodes are numbered from 1 to *n*. Next, each "Unknown" node is examined to identify the Unhealthy nodes that may cause the node's non-healthy state. The algorithm's output is one list per "Unknown" node that contains the "Unhealthy" node(s) identified as the root cause for the respective node's performance issues. The workflow described above is structured in two parts of the algorithm:

- Part A determines the status of individual elements in the network ("Healthy", "Unhealthy", "Unknown").
- Part B creates the Root Cause lists for the Unknown" nodes.

After both stages of the diagnostic process have been completed for a vector, and an anomaly or degradation has been successfully identified, with or without a valid root cause, a message containing information regarding the detected event along with the information regarding the relevant service/node is propagated to the Function Placement component.

Functions Placement component

This component comprises two sub-modules, one implementing the optimization function responsible for deciding on the optimal placement and a controller responsible for interfacing with the orchestrator in order to apply the deployment decision on the corresponding service/component. For the first module, typical function optimization techniques have been used to construct a multivariate function that takes into consideration all the possible variables of interest for this problem. The currently selected variables are:

- number of services/tasks;
- number of available infrastructures (across the available domains);
- computational load and maximum computational load of each infrastructure;
- the power cost of utilizing corresponding infrastructure;
- the computational cost of utilizing corresponding infrastructure;
- communication cost between related infrastructures.

Utilizing information retrieved from the available infrastructure and monitoring the deployed and available services/components, this module can provide optimal placement for targeted services. Specific details for each service are retrieved from the Service registry component. The placement algorithm is also configurable regarding the targets that are prioritized for the placement. Energy efficiency, the minimum number of infrastructures used, minimum amount of network traffic (focus on collocating as many services as possible) are some of the potential optimization targets. For the second module, an API has been developed and added as a wrapper to the first module in order to allow interfacing with the optimization algorithm and propagating the decisions to the orchestrator.

After a service is registered, the FP component is triggered in order to decide the placement for it. This initial placement also follows the selected policy for the placement, mentioned earlier. In this case, the initial placement of the industrial automation service's components is performed based on the restrictions regarding the placement of these components. This means that the Cloud management service is placed in the Cloud, the controller service is placed at the Edge, and the Autonomous robot agents and service roles are placed on the robots. When a need to find a new placement for one or more of these components arises, the FP is triggered to decide on this new placement. During this scenario, the service roles of the robots are frequently required to be migrated due to injected impairments or artificial stress.

4.3.3 Scenario 4.3: Improving service downtime and reducing costs using Predictive Orchestration

This scenario is focused on moving from a reactive approach in handling unexpected events to a proactive one, based on predicting the future states of the services and resources in order to identify the conditions that will lead to said events.

4.3.3.1 Scenario description

This scenario aims to demonstrate how utilizing AI/ML enablers to predict the behaviour of services and/or components can lead to increased efficiency and reduced costs, operational and maintenance, of industrial operations and systems. These mechanisms rely on using monitoring data from selected services or components to train predictive models capable of identifying upcoming critical events. The objective of this scenario is to demonstrate how the deployment and utilization of these AI/ML enablers could accomplish the proposed improvements for the operational workflows inside an industrial context. Moreover, these mechanisms are used to show how the impact of unexpected situations can be minimized in an automated way without requiring immediate human intervention. Besides the implementation itself, this scenario demonstrates the benefits of utilizing AI/ML enablers that rely on monitoring data from the infrastructure and deploying services in order to optimize the orchestration of services and allocation of roles/tasks in an industrial context. In this way, the target KPIs/KVIs can be monitored, and in extreme cases, corrective actions can be taken pre-emptively before a critical event occurs. For the demo implementation, the focus is on the health and power consumption of AGV components like batteries and motors, shown in Figure 4-13. The behaviour of those components is monitored in order to predict upcoming critical events such as malfunctions, overvoltage, extreme stress, low power, etc., and trigger the appropriate orchestration actions to avoid/handle them pre-emptively. This scenario utilizes all the software components introduced in the two previous scenarios and extends them with the introduction of the predictive orchestration component.

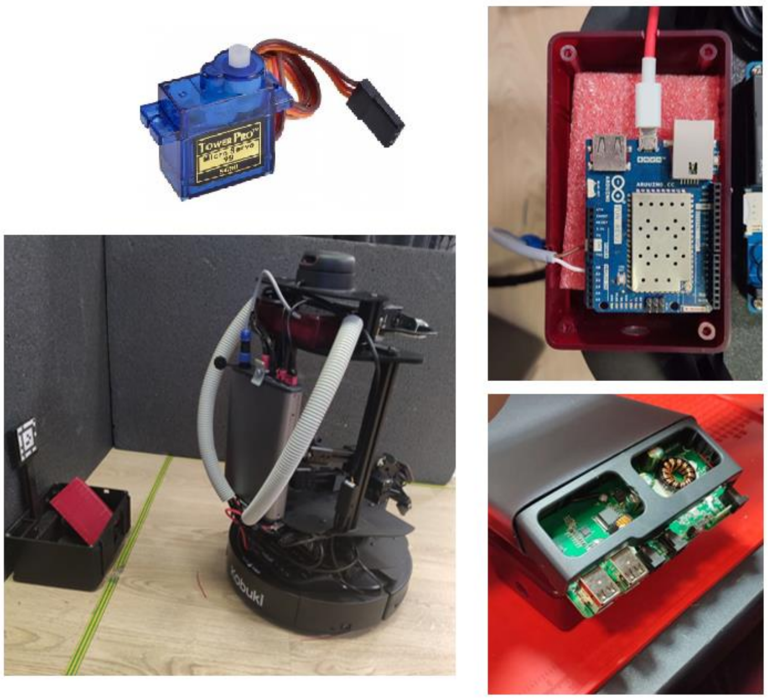


Figure 4-13. Monitored components.

4.3.3.2 Software components

In this Scenario 4.3, all the components developed for Scenarios 4.1 and 4.2 are reused, and the set of functionalities is extended by introducing a new component, the Predictive Orchestration component, seen in Figure 4-3, tasked with handling the prediction of future states of the deployed services and infrastructure.

Predictive Orchestration component

The Predictive Orchestration component has been developed with the purpose of predicting the behaviour and performance of monitored services and nodes across the available infrastructure. The purpose of these predictions is to compile an image of the future state of the infrastructure and deployed services. To accomplish that, the component utilizes AI/ML models specifically for time series forecasting. This future state is used by the functionality allocation component in order to discern if there is a need for pre-emptive actions to prevent upcoming critical events. These actions increase the efficiency of the industrial scenario operations by reducing the maintenance costs, reducing the time to replace malfunctioning workers, reducing the time stalled/blocked time between stopped operations, and in general, reducing industrial operations downtime while also minimizing maintenance costs by scheduling it proactively. Since the context of this demo is that of an industrial environment and its automated operations, appropriate pre-trained models for the specific metrics that will be taken into consideration will be utilized to enable the prediction of the behaviour of the various AGV components, like the battery, the motors, etc.

4.3.3.3 Functional behaviour

This scenario relies on the already provided functionalities from Scenarios 4.1 and 4.2 and introduces a new workflow on top of them. The predictive orchestration, an extension of the closed-loop M&O process as described in Scenario 4.3 (Subsection 4.3.3), is demonstrated. The purpose of these operations is to be proactively triggered in order to prevent an undesired event, like performance degradation and malfunctions, from happening. This sequence diagram is shown in Figure 4-14. A description of the new component's functional behaviour can be found in the following subsections.

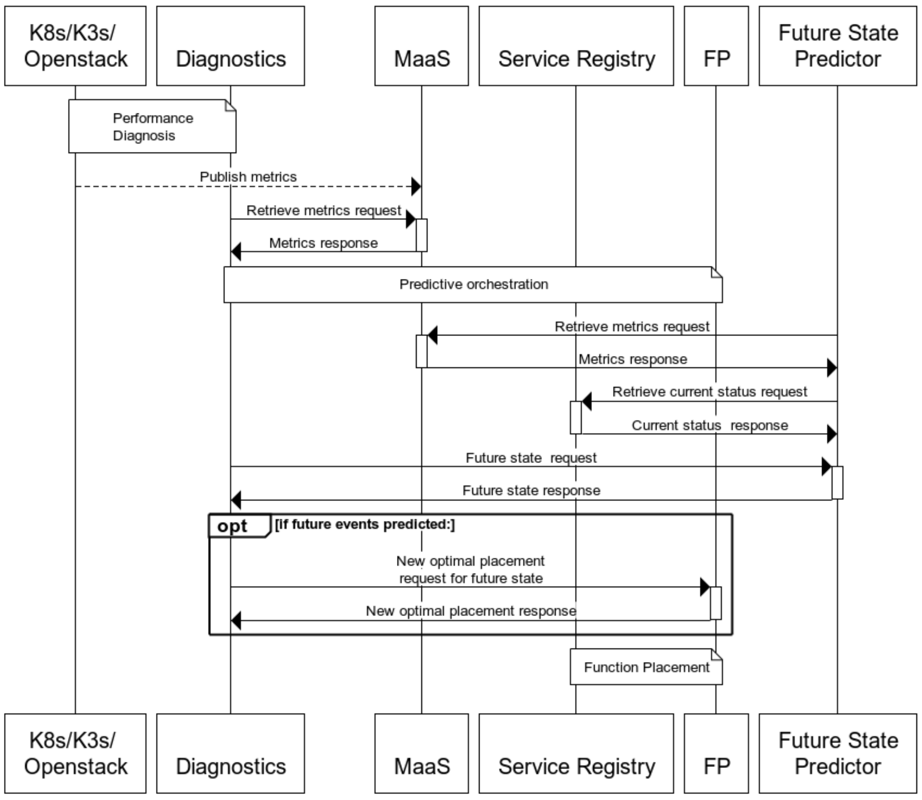


Figure 4-14 Predictive orchestration workflow

Predictive Orchestration component

The prediction functionality of this component is implemented using multiple AI/ML algorithms for trend detection, univariate and multivariate prediction, and prediction evaluation. These algorithms, similar to the ones of the Diagnostic component, have been developed as external modules to be used in conjunction with the predictive orchestration component, thus allowing its extension with others as well. The currently implemented algorithms are Facebook's Prophet [FPR], ARIMA/SARIMA [ARI], and Univariate and Multivariate Linear Regression [MLR]. These models are used individually, mainly depending on the type of metric, if it is additive, etc., and the predictions it provides. The predictions of each model are cross-compared and weighted against the actual observed value for which they provided predictions. This acts as a self-validating feedback loop. Greater weight is assigned to the models that give more accurate results, while the rest of them are assigned lower weights and might also have their hyperparameters tuned in order to provide better future results. For this component, an API has also been developed for interfacing with the rest of the components and external management. During the operation of the service, its state is monitored and analysed by the MaaS and Diagnostic components, respectively. In parallel, the Predictive Orchestration component utilises the aforementioned models to predict the future state of the monitored metrics and KPIs. These predicted values are fed back to the Diagnostic component in order to analyse the predicted future state for any anomalies. When an anomaly is detected in that future state and verified as valid through the mechanisms described earlier, the FP component is then proactively triggered to decide the optimal placement of the service components to prevent the predicted anomaly.

4.3.4 Scenario Deployments

The various developed software components are deployed on resources on the cloud and edge domains. The deployment layout is pictured in Figure 4-15.

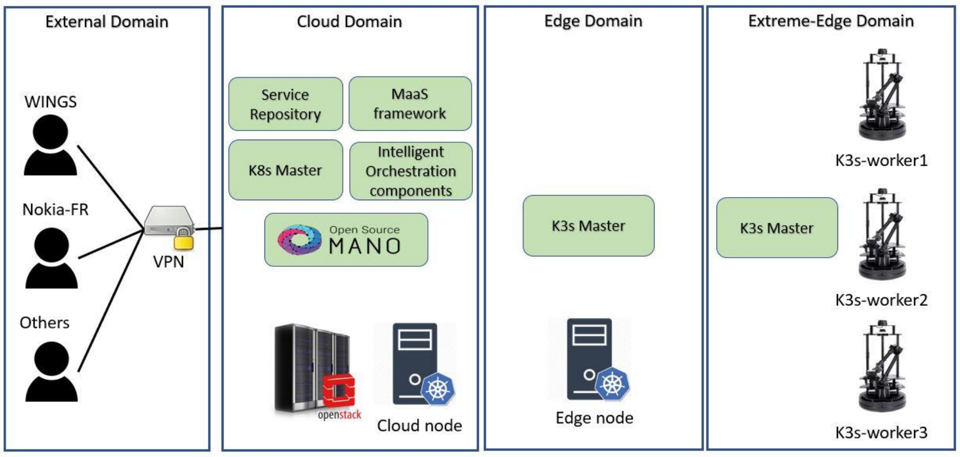


Figure 4-15. Demo #4 deployment overview.

This deployment considers a common orchestrator for the extreme-edge nodes, K3s, which is available, meaning that its connection information is already provided, a priori to these nodes, for their management and orchestration. In cases where such an assumption is not made, another solution to the problem of extreme-edge discovery will need to be implemented, like the one proposed in the complimentary lab experiment regarding node discovery in subsection 6.2. The communication between the extreme-edge nodes, the AGVs, and the rest of the infrastructure is implemented using a commercial 5GNR solution, the Quectel RM500Q modem [QUEC] and utilizing an overlay VPN network over the commercial 5G network. The communication between the various Cloud-Edge domain components is handled by wired connections and virtual networks. The interaction between different users and the robots in this demo is implemented utilizing a Virtual Reality (VR) headset and a laptop. Through these devices, the user can view the Digital Twin application and also interact with the individual robots or configure/control the service overall. A large monitor is also used to provide the overall presentation of the Digital Twin application for other users that are not currently interacting with it. A secondary monitor is also used to show the status of the deployment in the form of metrics and KPIs collected and ingested by the system through the MaaS framework. Other than the infrastructure requirements of the deployment, the demo also requires a minimum amount of space for the robots to localize and move inside. This space needs to be mapped as the emulated, or the actual, if that is the case, industrial floor space. This space should be a minimum of 9 square meters up to no known size limit. For this demo specifically, the used area is 12 square meters, i.e., 4x3 meters. Finally, Table 4-1 contains the compute resources allocated for each of the demo components.

Table 4-1. Demo #4 deployment resources.

Host	**Description**	**Type (VM/ Physical)**	**OS**	**Arch.**	**CPU (#)**	**RAM (GB)**	**Disk (GB)**
MaaS framework	MaaS framework host	Virtual	Ubuntu 20.04	x86_64	8	32	80
K3s-master	K3s master for the edge	Virtual	Ubuntu 20.04	x86_64	4	4	100
K3s-master	K3s master for the extreme-edge	Virtual	Ubuntu 20.04	X86_64	4	4	80
K8s-master	K8s master for the cloud nodes	Virtual	Ubuntu 20.04	X86_64	4	8	80
OSM	Service orchestrator	Virtual	Ubuntu 20.04	x86_64	2	4	80
Intelligent orchestration components	Service Registry, Diagnostic, Predictive mechanism, Functions placement	Virtual	Ubuntu 20.04	x86_64	4	4	100
Service Repository	Orchestration repositories for docker images and helm charts	Virtual	Ubuntu 20.04	x86_64	2	4	100
K8s cloud node	Cloud worker node	Virtual	Ubuntu 20.04	x86_64	4	8	80
K3s Edge node	Edge worker node	Virtual	Ubuntu 20.04	x86_64	4	8	80
k3s-worker-1	Robot worker	Physical	Ubuntu 20.04	x86_64	4	8	256
k3s-worker-2	Robot worker	Physical	Ubuntu 20.04	x86_64	4	8	256
k3s-worker-3	Robot worker	Physical	Ubuntu 20.04	x86_64	4	8	256

5 Demo #5: Data-driven device-edge-cloud continuum management

5.1 Demo overview

Demo #5 is directly related to Hexa-X Objective 3 (Connecting intelligence towards 6G), which targets turning AI/ML into an essential component of the B5G/6G technology. Specifically, for WP6, this objective is intended to develop AI/ML-powered enablers for orchestration and service management in order to achieve higher efficiency and enable new services (and revenue streams) considering an expanded focus by targeting the continuum from the end-devices to the cloud/core networks. Demo #5 integrates four different scenarios that are executed in a simulated urban environment:

- **Scenario 5.1:** Continuum orchestration of AI/ML-driven traffic light control service. Broadly, the scenario consists in a simulated road-traffic urban environment where traffic lights are controlled by a network service which is deployed on the device-edge-cloud continuum, and that relies on an AI/ML algorithm to increase the mobility of the vehicle in the simulated environment. Although the urban scenario is mostly simulated, the traffic lights have been implemented through a set of real LED lamps that are connected to a cluster of low-power computing nodes (a Raspberry Pis cluster) running the traffic lights control functions, which in practice constitutes a realistic extreme-edge implementation for this demo. The service functions are deployed on demand by a service orchestrator able to manage resources on this extreme-edge environment and on other regular computing nodes simulating the edge and the cloud domains.
- **Scenario 5.2:** Prediction-based URLLC service orchestration and optimization. This scenario is a particularization of the simulated road-traffic urban environment in the previous Scenario 5.1, but with a small set of vehicles, where the vehicles request URLLC-type services. In this case, an AI/ML data-driven approach is used to predict the road traffic increases and proactively scale-up the needed resources to make the necessary servers available before the service quality actually drops. Also, if possible, computational tasks can be offloaded to the extreme-edge resources, i.e., close to the end-users.
- **Scenario 5.3:** Reactive security for the edge. Relying on the main concept from the previous scenarios, this scenario targets to showcase how M&O security can be enforced across the whole network (from the extreme-edge to the central cloud), also targeting specific network domains and network slices. The scenario covers various aspects:
  - The prediction of the future state of a device. In this case, the M&O security framework is used to proactively detect the early steps of an attack and take actions to block the predicted next steps of the attack.
  - Ensuring the lowest possible latency. In this case, the security orchestration framework is distributed to improve the time-to-detect and time-to-remediate KPIs.
  - Re-locating service as close as possible to the end-user device. In this case, the scenario shows how security functions can be distributed in a very granular way, down to the extreme-edge domain devices, relying on a hierarchical organisation of security orchestrators. This approach allows deploying local autonomic lightweight orchestrators on the edge and complex orchestrators in the core.
- **Scenario 5.4**: MLOps techniques to deploy AI/ML service components. This scenario approaches the MLOps methodology to deploy and operate an AI/ML model on a simulated MNO infrastructure. It highlights the cooperation between two stakeholders involved in the MLOps workflow: the MNO (which operates the AI/ML model on its infrastructure) and the SW Vendor (which designs and trains the AI/ML model). Aligned with the Design Layer concept introduced in [HEX22-D62], this cooperation between two independent entities is one of the main concepts explored in this Demo #5 since it implies the sharing of the training data from the MNO to the Vendor, which in real-life scenarios could involve the sharing of sensitive data (e.g., personal data) between the two entities to train the AI/ML model. The scenario also explores other well-known concepts in the AI/ML scope: the drift management concept, considering how to integrate this in the MLOps workflows.

In the following sections, each of these scenarios is described in more detail.

5.2 Innovations related to the demo

Although in a reduced scope, this Demo #5 addresses most of the innovations described in Deliverable D6.2 – Section 5.3 [HEX22-D62], namely:

- **Unified orchestration across the "extreme-edge, edge, cloud" continuum.** This is the main topic addressed in Scenario 5.1 - integrating the traffic lights infrastructure at the extreme-edge domain, together with the edge and the central cloud components. Also, in Scenario 5.2, where the same prediction algorithm that drives the orchestration decision can be used for the three different resource domains: depending on the time required to activate each of them, a different prediction window can be employed.
- **Unified management and orchestration across multiple domains, owned and administered by different stakeholders.** This is partially addressed in Scenario 5.4, where MLOps techniques involve both: the MNO domain and the SW Vendors domain.
- **Increasing levels of automation.** This innovation is addressed in all the Demo #5 scenarios. In Scenario 5.1, RL algorithms contribute to automatically operating the traffic lights' status in a more efficient way. In Scenario 5.2, an LSTM [LST] network prediction already determines the number of resources to be (de)activated over time. In Scenario 5.3, layered closed loops are used to automatically detect, contain, and eradicate a cyber-security threat (the level of automation depends on the authorizations granted to the system since some remediation actions may require approval from human administrators). Finally, in Scenario 5.4, MLOps techniques are used to automatically train and deploy the AI/ML models on a simulated MNO infrastructure, reducing manual intervention for both: the AI/ML models vendor and the network operator.
- **Adoption of data-driven and AI/ML techniques in the M&O system**. Data and AI/ML techniques are used to feed a Reinforcement Learning model in Scenario 5.1. Also, in Scenario 5.2, they are used to compute the number of servers required based on the real-life traces from an Italian city [MKV+22]. Also, in Scenario 5.3, although no AI/ML technique is currently proposed in the scenario itself, the proposed system is evolutive, meaning that it is able to integrate AI/ML modules to leverage its automation capabilities. In this sense, AI/ML could be useful both: in the detection/analysis phase and in the remediation phase. Finally, in Scenario 5.4, data is used to train the AI models that need to be generated at the SW Vendor side and to monitor the AI models once deployed on the MNO infrastructure.
- **Adoption of the cloud-native principles in the telco-grade environment**. Cloud-native principles are applied in two main ways within this Demo #5: (i) by using micro-services for implementing the different software components in the demo in the form of Docker containers, and (ii) by showcasing mechanisms for the services to be deployed and updated using DevOps practices, implementing continuous integration and continuous delivery (CI/CD) pipelines with a high automation degree. This is specifically addressed in Scenario 5.4.

5.3 Demo implementation

As for the previous Demo #4, the implementation of this Demo #5 has been designed in such a way that it can be aligned with the M&O architectural design introduced in the previous Deliverable D6.2 [HEX22-D62]. In the following subsections, the scenarios considered for this demo are presented according to the same structure used for the previous Demo #4, describing the different components used for each scenario, its functional behaviour, and the deployment details for each.

5.3.1 Scenario 5.1: Continuum orchestration of AI/ML-driven Traffic Lights Control Service

5.3.1.1 Scenario description

Scenario 5.1 aims at demonstrating how the 6G technology can be used to improve the road traffic flow in urban environments by controlling the traffic lights using AI techniques. The objective is to demonstrate how the deployment of this AI/ML-enabled control could improve road traffic mobility compared to the common approach, i.e., the traffic lights activation based only on deterministic and periodic time patterns. Beyond the implementation itself, what this scenario suggests is the possibility of deploying more advanced sets of connected traffic lights to enable more efficient road traffic control strategies. These traffic lights, beyond simply switching on/off their lamps, would also be enabled to perceive their immediate environment (e.g., through cameras installed on the traffic lights themselves or in nearby locations) and would send this information to the edge application through the 6G network. Here, based on AI/ML algorithms, it could trigger more intelligent actions by adapting the traffic lights switching times to the actual traffic conditions (see Figure 5-1). This would help to reduce traffic jams and minimize waiting times, as well as provide more advanced functionalities, such as coordinating the traffic lights activation to give preference to priority vehicles passing through (e.g., ambulances or other emergency vehicles), among others. Besides, real-life implementations of a system like this could help to reduce CO_2 emissions in polluted cities, which is also in line with the *Sustainability* KVI defined in [HEX21-D12]. For the demo implementation, a simulated set-up has been devised[2], consisting of:

- The extreme-edge components are implemented by a set of hardwired red/orange/green LED lamps representing the set of traffic lights to be controlled, which are connected to a set of small-scale Raspberry Pi computers running the low-level traffic light control processes.
- The edge cloud, where the AI/ML model is used to control the traffic lights, is trained and executed. Specifically, the classical Q-Learning algorithm [Wat89], a Reinforcement Learning based algorithm, has been used for this[3]
- The central cloud, where a so-called central urban mobility server and other core NFs are be hosted.

For the vehicle's flow generation, a well-known open-source framework for the simulation of urban mobility scenarios has been used (SUMO [LBE+18]).

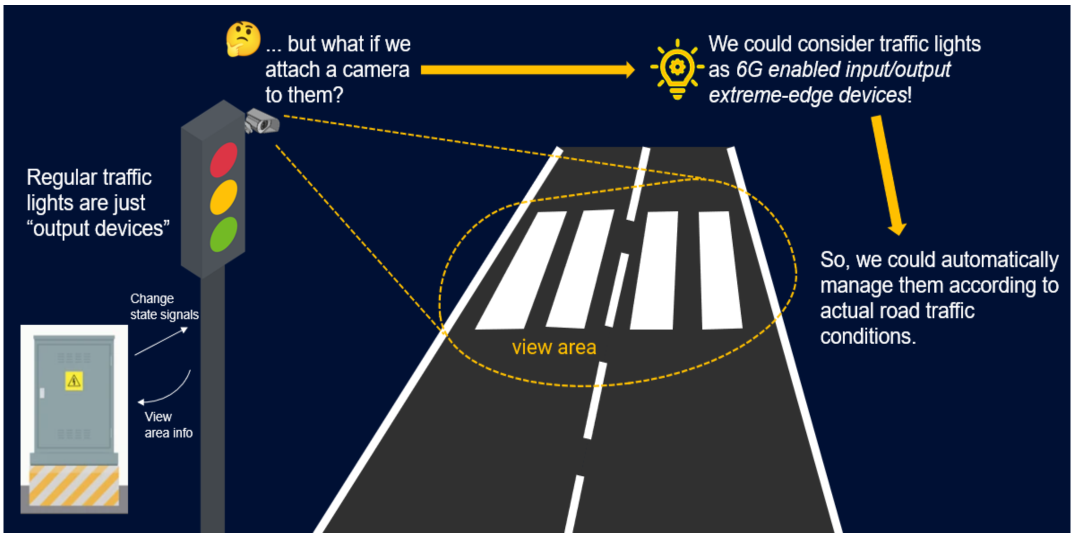

... but what if we attach a camera to them?
We could consider traffic lights as *6G enabled input/output extreme-edge devices!*
Regular traffic lights are just "output devices"
Change state signals
View area info
view area
So, we could automatically manage them according to actual road traffic conditions.

Figure 5-1. Smart traffic lights to enable AI/ML-driven control.

This tool allows the generation of different traffic patterns with different kinds of vehicles over a simulated urban environment consisting of several intersections regulated by traffic lights. It also offers a GUI which displays in real-time the vehicle's flow and the traffic light's activation (see Annex I). However, although software-based, this simulated environment is in full synchronisation with the actual LED lamps at the extreme-edge mentioned above (see Figure 5-2). More specifically, regarding the urban environment setup, a layout with four crossroads has been designed, with a traffic light at each branch of each crossroad. Three crossroads have four

[2] "Technology Readiness Level (TRL) [Hor20] for this demo is TRL 3 (experimental proof of concept).

[3] Please, be aware that for this demo scenario the main concern here is not on the AI/ML topic itself, but the demonstration of the continuum M&O concept including the extreme-edge domain. AI/ML is used here anyway as a way to align the scenario with what could be a use case for future 6G networks, where services based on AI techniques are expected to be more and more relevant.

branches, while the last one has only three, having a total of 45 individual lamps to be controlled (see Figure 5-3).

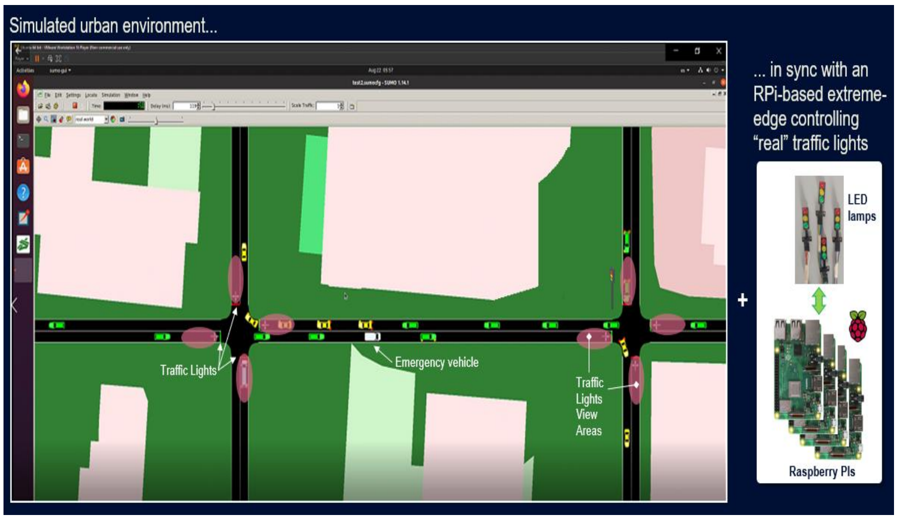

Simulated urban environment...
Traffic Lights
Emergency vehicle
Traffic Lights View Areas
... in sync with an RPi-based extreme-edge controlling "real" traffic lights
LED lamps
Raspberry PIs

Figure 5-2. Simulated urban environment synchronised with the *real* traffic lights.

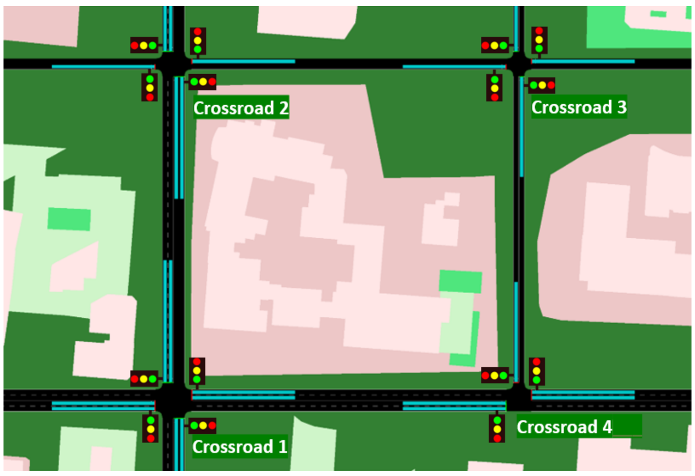


Figure 5-3. Scenario 5.1 – streets layout.

This setup is used to run the software simulation on the computer. However, as mentioned, a physical replica of this setup has also been built with real LED lamps, which are connected to four Raspberry Pi cards (one per crossover) to implement a realistic extreme-edge set-up. Figure 5-12 in Section 5.3.1.4 shows a picture of this replica. The traffic lights view areas (one per traffic light) are simulated within this scope, i.e., no real cameras have been deployed to measure the traffic density in each traffic-light control area. For the sake of simplicity, this functionality has been delegated to the above-mentioned SUMO simulator, which monitors those lane areas highlighted in blue in Figure 5-3[4]. The generation of the vehicles participating in the simulation is also delegated to SUMO, which generates them with specific trajectories and speeds on the given street layout. It progressively adds vehicles following a statistical distribution based on a random seed. Of course, the challenge to the AI is to control the traffic lights status in such a way to allow the vehicles to make their journey as quickly as possible, minimising (or even avoiding) waiting at red lights. More details about the demo implementation are provided in the following

[4] As previously mentioned, in a real-life application, this could be done by real video camera attached to the traffic light, together with an AI/ML-based image recognition software trained to detect high traffic density situations.

sections, starting with the main functional components (section 5.3.1.2), the overall functional behaviour of the demo (section 5.3.1.3), and the deployment details (section 5.3.1.4).

5.3.1.2 Software components

Figure 5-4 shows the main software components that have been used to implement this demo scenario, grouped in the three network domains: cloud, edge, and extreme-edge. Also, and in line with the M&O architectural design introduced in [HEX22-D62], the red dashed line on top groups the components in the M&O scope, i.e., the Extreme-edge Orchestrator, the Edge Orchestrator, and the Cloud Orchestrator, which handle the management of the resources in the three domains, with the Vertical Slicer, deployed in the cloud domain and responsible for E2E service management.

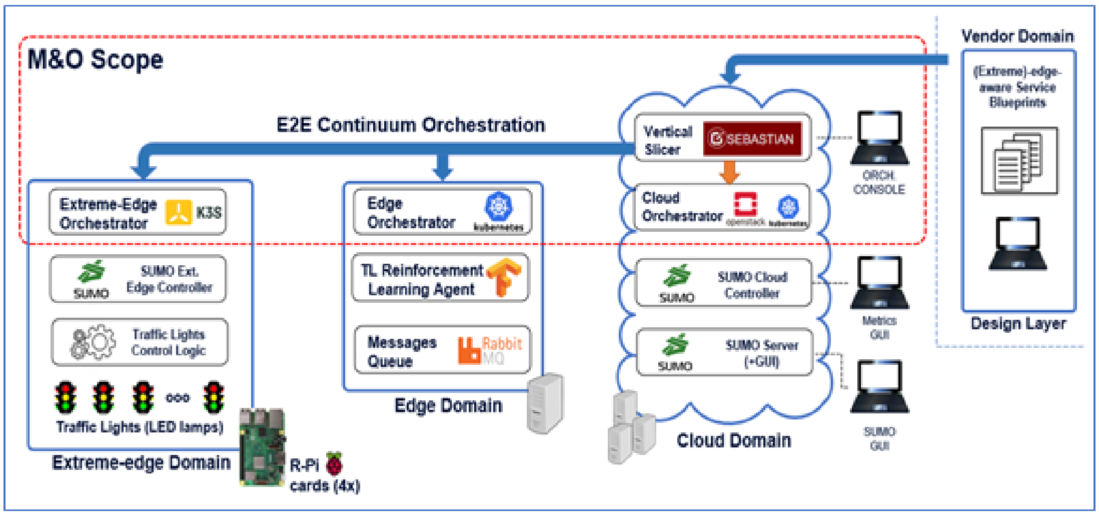


Figure 5-4. Main functional components of Scenario 5.1.

The Vertical Slicer has parent-child relationships with the rest of the domain-specific orchestrators. Altogether, these four orchestrators implement the E2E Continuum Orchestration function, which is the main topic addressed in this demo scenario. On the other hand, those components below the red dashed line are the service-related components, i.e., those components implementing the AI/ML-driven traffic lights control use case introduced in the previous section. In practice, this traffic lights control service is distributed over multiple containers, which are deployed dynamically by the E2E Continuum Orchestration function that operates at the Network Layer of the architecture and has been implemented explicitly for this demo. The Vertical Slicer (called SEBASTIAN), which extends an existing open-source software for the management of communication services and network slices in 5G networks [5GR21-D24], interacts with various virtual infrastructure platforms to manage computing resources across the continuum of the extreme-edge nodes (represented by_Raspberry Pi nodes connected to the traffic lights for their control), edge nodes, and cloud nodes. This resource continuum, which is mapped into the *Infrastructure Layer* of the Hexa-X M&O architecture, is handled through different virtual infrastructure platforms. Specifically, a K3s [K3S] cluster is used for the management of the four extreme-edge nodes, K8s [KUBa] for the management of edge and cloud resources for container-based deployments, and OpenStack [OST] for the management of cloud resources for VM-based deployments. Although this scenario does not explicitly consider 5G connectivity, experiments with the SIMU5G NR simulator [NSS+20] working with the traffic related to this scenario are reported in Section 6.3. In the following subsections, the software components developed for this scenario in each domain are described in more detail.

<u>Extreme-edge domain</u>

The following are the functional components in this domain:

- **Traffic Lights Control Logic**. Generally speaking, this is basically a sort of *low-level driver* in charge of activating/deactivating each physical traffic light led in the set-up. It receives commands from the AI/ML agent (running on the edge domain) and parses them to properly activate/deactivate the traffic lights (through the *General-Purpose IO -GPIO-* ports in the Raspberry Pis), according to the AI/ML agent requests. However, although this is its primary function, it also implements an additional low-level control logic to avoid certain undesired situations that the AI/ML agent might lead mainly during its training stage. These low-level control logic functions include but are not limited to preventing lights from blinking excessively often, or the opposite, never changing their status (they could remain red or green for too long, or even all the time), and also preventing from inconsistent situations, such as all lights in a crossroad being either red or green at the same time. In other words, this control module, besides just activating/deactivating the traffic lights themselves, also performs a kind of "hard-wired" filtering function, preventing the AI from provoking obvious inconvenient situations. From the AI agent's perspective, the rules set is part of the "external" environment from which it learns, so these rules are integrated into the learning process. This component has been developed specifically for this scenario in Demo #5. Four instances are deployed (one on each of the Raspberries) in the form of K3s pods to control each of the crossroads in the setup (i.e., each instance controlling one crossroad). The implementation uses *Python 3.8.10* and is optimised for *ARM64*-based architectures.
- **SUMO Extreme-edge Controller**. This component, which works as a complement to the *SUMO simulator*, can be understood as a kind of *kernel* in terms of synchronising all the other components in the simulation process, as well as a data-collector from the *SUMO* simulation that gathers data and interacts with the simulator. The component has been implemented from a "containerised" perspective to be deployed as a K3s pod. Four instances are allocated, one on each Raspberry Pi device, to collect data individually from each crossroad at a time. This functional approach has been used in order to ease the implementation of a distributed E2E architecture. It has been developed using *Python 3.8.10,* optimised for *ARM64* architectures. The component was developed mainly because of the usage of the SUMO Simulator in this Scenario 5.1 requires an interface operating through its TraCI API [LBE+18] in order to obtain real-time data from the ongoing simulation. Its main functionalities are:
    - **Components' Synchronisation**. Proper synchronisation between the *SUMO Server* and the rest of the functional components must be assured in order to have a correct simulation behaviour. To get this, a specific *SYNC* signal is sent from each *SUMO Extreme-edge Controller* instance before a new simulation step starts.
    - **Data Collection:** It collects the data required by the *AI/ML* function on each crossroad (i.e., the number of vehicles). The data is retrieved using the values obtained through each crossroad's LADs and is used to comprise the information sent to the *AI/ML* functional component.
    - **Data Sharing:** The required data is sent to the *AI/ML* component as a structured Python data dictionary.
    - **Data Reception:** This functional component expects data from the *Traffic Lights Control Logic* component comprised of traffic lights status updates.
    - **Update the traffic lights simulation statuses.** Traffic lights status updates are sent to the *SUMO Server* so that the actions that result from the *AI/ML* component are applied to the simulation scenario.
    - **Extreme-edge Orchestrator**. This is the orchestrator used to perform the life-cycle management of the components at the extreme-edge, i.e., the Traffic Lights Control Logic and the SUMO Extreme-edge Controller previously described. As can be seen in Figure 5-4, it has been implemented using K3s, a certified lightweight K8s distribution built for IoT and Edge computing [K3S]. This orchestrator interfaces with the overall Vertical Slicer component in the cloud in order to integrate it into the continuum orchestration workflows. This interaction is based on the K8s API (exposed by the K8s API server within K3s) to retrieve

the list of extreme-edge nodes in the cluster, subscribe for and retrieve notifications on cluster events (e.g., new nodes), and perform CRUD operations on namespaces, deployments and pods for service provisioning. K3s has been deployed in the form of a lightweight K8s cluster, with distributed storage, on the four available Raspberry Pi nodes, with one of them acting as the master node and the other three as workers. All the nodes have been labelled accordingly in order to assign one crossroad to each Raspberry Pi so that the K3s deployments can be allocated correctly.

Edge domain

The edge domain hosts the following components:

- **The Traffic Lights Reinforcement Learning Agent**. This is the *brain* of the service. It is based on the Q-Learning algorithm [Wat89], a model-free algorithm that aims to learn the value of an *action* based on a particular *state* depending on the *rewards* it receives from its environment. In the context of this demo, the environment is the urban environment described in the previous paragraphs (see Figure 5-5), where its overall *state* at a certain moment is defined by the traffic situation in the crossroads (taken from the monitored lane areas nearby the traffic lights), while the actions are the commands sent to the traffic lights to put them in a specific state (red, green, or yellow). The *reward* is a value based on the average speed and average waiting time of vehicles: a higher average speed will produce positive rewards, and higher average waiting times will produce lower or even negative rewards, *reinforcing* the agent to learn which specific *actions* on specific *states* produce higher *rewards*.

  The AI/ML application developed for the demo consists of four independent RL agents, each acting on the traffic lights of a specific crossroad. However, these four agents are deployed together on a single AI/ML application instance, on which each agent receives information on all the LADs in the four crossroads. This is intended to provide each agent with overall visibility since the traffic situation in a specific crossroad could be affected by what could be happening in the neighbour crossroads also. This way, learning of each agent is not only based on what could be happening locally but also on a broader level.

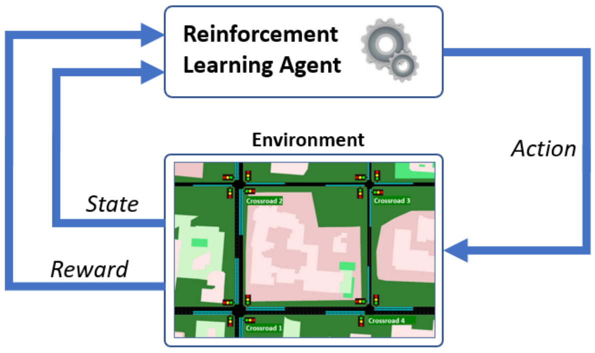


Figure 5-5. Reinforcement Learning in Scenario 5.1.

  The AI/ML application has been implemented using Python language and some additional libraries like Pandas to handle the Q-Table and NumPy to process input data. More information explaining how this agent performs its work is provided in subsection 5.3.1.3 (Functional Behaviour) and also in Annex II.

- **Messages Queue**. This is an instance of the well-known open-source RabbitMQ messages broker [RAB]. In the context of this demo, it is used to establish communication between the software components of the demo. More specifically, to communicate the RL Agent with the SUMO Extreme-edge Controller and the Traffic Lights Control Logic components at the extreme-edge (see Section 5.3.1.3).
- **Edge Orchestrator.** This is an instance of the well-known open-source K8s system [KUBa] for automating the deployment, scaling, and management of containerized applications. It plays the same role as the K3s orchestrator at the extreme-edge explained

before, but in this case, for the applications on edge. It also interfaces (using the K8s API) with the overall Vertical Slicer in the cloud in order to integrate the edge components with the continuum orchestration workflows. It has been deployed in the form of a Rancher K8s Engine (RKE) v2 multi-node cluster with distributed storage and a service mesh based on Istio [IST]. A dedicated VM has been created to act as the K8s controller and two worker nodes within the *edge domain.* This orchestrator is also part of the compute continuum M&O and, therefore, it interfaces with the *Vertical Slicer* component to enable the same operation available for the *extreme-edge* orchestrator.

Cloud domain

As can be seen in Figure 5-4, the cloud domain contains the following components:

- **SUMO Server.** This is the main component of the above-mentioned open-source urban mobility SUMO simulator [BBE+11]. SUMO has been chosen for implementing this scenario due to its wide-range of real-time interaction, visualisation, and metrics extraction features in the urban scenario devised for this demo. Specifically, SUMO can be used to simulate the movement of a number of individual vehicles inside the virtual roads network designed for the demo, taking into account things like speed restrictions, traffic lights, and the presence of other vehicles. SUMO is also able to provide metrics reporting on the overall state of the vehicular scenario (i.e., mean travel time metrics, mean waiting time metrics, vehicle consumption metrics, vehicle pollution metrics, etc.) [LBE+18]. The simulator also includes a GUI (accessed thru this SUMO Server component) that allows the user to visualise the SUMO simulations and watch the behaviour of the vehicles under the running conditions (in a layout like the one in Figure 5-3). In addition to the simulation capabilities, SUMO also provides a complete API (the so-called Transport Control Interface - TraCI) [LBE+18]. The TraCI API allows SUMO users to programmatically control and interact with the SUMO simulations enabling the integration of SUMO with other tools and platforms. This is, of course, a key aspect that has been used in this demo in order to integrate the simulator with the other components in the edge and the extreme-edge domains. Specifically, the TraCI API has been used in the demo to, of course, interact with the traffic lights and to get real-time information about the number of vehicles in the lane area detectors associated with the traffic lights. The SUMO Server component has been deployed, in a dedicated VM on the *cloud domain*, as part of an OS-package in the form of a C++ binary. A computer, acting as a client, can be connected to this SUMO Server component to allow the end-user to launch the simulations and visualise them in real-time.
- **SUMO Cloud Controller**. This component is in charge of loading the simulation configuration parameters, vehicular data-flows and managing the initial SUMO GUI appearance. Besides, it also implements the features required to gather different real-time metrics (i.e., number of running vehicles, stopped vehicles, CO_2 consumption, fuel consumption, PMx, etc.) and to store them in a database so that they can be monitored and plotted in several monitoring stacks (e.g., Grafana [GRA]). This component has been developed specifically for this demo using *Python 3.8.10*.
- **Cloud Orchestrator**. This component is with a similar function to the one of the Extreme-edge Orchestrator and the Edge Orchestrator described in the previous sections. However, here, its function is to specifically orchestrate the components in the cloud domain, i.e., the SUMO Server and the SUMO Cloud Controller. In this case, these components have been implemented by means of virtual machines (instead of containers, as in the previous edge and extreme-edge domain), so the Cloud Orchestrator has been implemented using OpenStack Kolla (version Yoga) [YOG] for managing these VMs. Like the other orchestrators in the other domains, this one also interfaces with the overall Vertical Slicer Orchestrator using the OpenStack REST APIs for the dynamic creation and management of virtual networks, subnets, and VMs (in particular, relying on the subset of Compute API and Networking API v2.0 exposed by OpenStack and securely accessed through an OpenStack client library integrated with the Vertical Slicer).
- **Vertical Slicer**. This component implements the overall E2E continuum orchestration function. It is an evolution (specifically implemented for this Hexa-X project) based on

the open-source Vertical Slicer software [5GR21-D24], developed by Nextworks in the context of previous 5G-PPP and ESA projects for the management of vertical services in 5G networks, their mapping to end-to-end network slices, and the lifecycle management of these network slices across RAN, core and transport domains. This is, of course, a core part of the demo since this is, in fact, the component implementing the *continuum management and orchestration* concept.

The original software was already composed by (i) a Communication Service Management Function (CSMF) handling the lifecycle management logic of vertical services, (ii) a Network Slice Management Function (NSMF) responsible for the provisioning and orchestration of end-to-end network slices, and (iii) some Network Slice Subnet Management Functions (NSSMF) specialized for the management of RAN, core and transport functions and resources. For implementing this Hexa-X Scenario 5.1, the CSMF has been extended to manage vertical services running in the extreme-edge, edge, cloud continuum, and the software has been complemented with a new component specifically dedicated to the resource orchestration over multiple edge/cloud platforms featuring extreme-edge nodes, as represented in Figure 5-6.

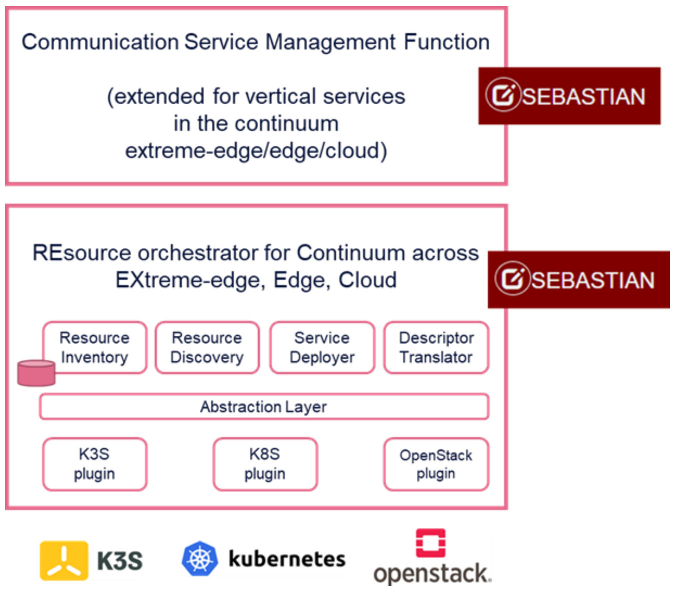


Figure 5-6. High-level software for orchestration in Scenario 5.1.

The new software component (REC-EXEC – REsource orchestrator for Continuum across EXtreme-edge, Edge, Cloud - the lower rectangle in the figure) has a modular architecture allowing it to interact with several platforms at its southbound, handling the details of their interfaces through an abstraction layer that exposes a unified interface towards the resource orchestration logic components. In this implementation, the abstraction layer works with three different plugins for K3s, K8s and OpenStack, respectively (those orchestrators deployed locally at each domain in the demo). The orchestration logic is implemented through four modules: Resource Inventory, Resource Discovery, Service Deployer and Descriptor Translator. The Resource Inventory and the Resource Discovery components are dedicated to the management of resource clusters and available nodes in the extreme-edge, edge and cloud domains (an additional experiment, specifically focused on the resource discovery feature, is reported in Section 6.2). The Resource Inventory works as a dynamic catalogue of K8s/K3s clusters and clusters' nodes, monitoring the events related to the composition of the clusters (e.g., the addition of a worker node, removal of a worker node, etc.) and exposing endpoints to:

- register a new cluster for monitoring;
- retrieve information and receive notifications of the clusters under monitoring;
- retrieve the information of the nodes of a given cluster;
- retrieve the information of the nodes of a given cluster that match a set of specified labels;
- unregister a cluster under monitoring.

The Resource Inventory works as a dynamic catalogue of K8s/K3s clusters and clusters' nodes, monitoring the events related to the composition of the clusters (e.g., the addition of a worker node, removal of a worker node, etc.) and exposing endpoints to:

- register a new cluster for monitoring;
- retrieve information and receive notifications of the clusters under monitoring;
- retrieve the information of the nodes of a given cluster;
- retrieve the information of the nodes of a given cluster that match a set of specified labels;
- unregister a cluster under monitoring.

In order to retrieve the information of the K8s-like clusters and be aware of any changes that can occur in terms of nodes that form the clusters themselves, the K8s/K3s clients are used to monitor the registered cluster making use of watch objects and API calls. The adopted library allows the Resource Inventory plugins to interact with all kinds of K8s clusters compliant with the specifications detailed by the K8s API (K8s API server). In the case of Hexa-X, the interaction is with K3s instances deployed in extreme-edge domains and with the standard *kubeadm* tool deployed in K8s clusters for edge and cloud domains.

The Service Deployer allows instantiating the various service virtual components in the underlying platforms, giving the possibility to specify the target nodes. The Service Deployer is supported by the Descriptor Translator, in charge of translating the platform-agnostic service descriptors received from the CSMF into the descriptor formats adopted in each platform. For example, for OpenStack, the original service descriptor would be translated into a Heat template. The CSMF has been extended with two new internal modules. A Resource Allocation component has been added to decide the target platform, clusters and nodes where the service components should be deployed. A driver towards the REC-EXEC allows for retrieving information about nodes and resources available in the continuum (to feed the Resource Allocation logic) and to trigger the instantiation and all the other lifecycle management actions of the service components on the underlying computing resources. For the demo, the CSMF and the REC-EXEC components are instantiated in two VMs running in the OpenStack cloud environment in Nextworks laboratory, interconnected to Atos testbed via VPN. Each VM hosts a set of Docker containers with the various modules of CSMF and REC-EXEC. Further details on the VMs' requirements are provided in Section 5.3.1.4. The CSMF service catalogue includes the service blueprints defining the characteristics of the traffic light control service components to be deployed and orchestrated in the infrastructure continuum. The blueprints have been defined manually, and they are expected to be the output of the service design layer in the overall service lifecycle. To this point, all the components that have been deployed for the implementation of the demo have been introduced. The following subsection, 5.3.1.3, describes how these components interact with each other to provide the required functionality.

5.3.1.3 Functional behaviour

This section describes how the different components in the previous section interact with each other to provide the demo functionality. Aligned with the split of Figure 5-4, this description is also split into two parts: the M&O functionality on the one hand and intelligent traffic control functionality (i.e., the managed service) on the other.

M&O functionality

From an orchestration point of view, both the CSMF and the REC-EXEC realise a workflow for the provisioning of the service. In particular, all the internal services of the REC-EXEC are involved: the Resource Inventory (RI), the Resource Discovery (RD), the Service Deployer (SD), and the Abstraction Layer (AL) with the specific plugins. Regarding the resources available, both Edge cluster (EC) and Extreme-edge Cluster (EEC) are considered in this workflow. Before the actual service provisioning, an initial configuration stage allows the collection of information about the resources available in both clusters and registers for any dynamic change. This approach is used to keep the Resource Inventory (RI) continuously synchronized with the nodes entering and leaving the clusters at runtime. The workflow is depicted in Figure 5-7.

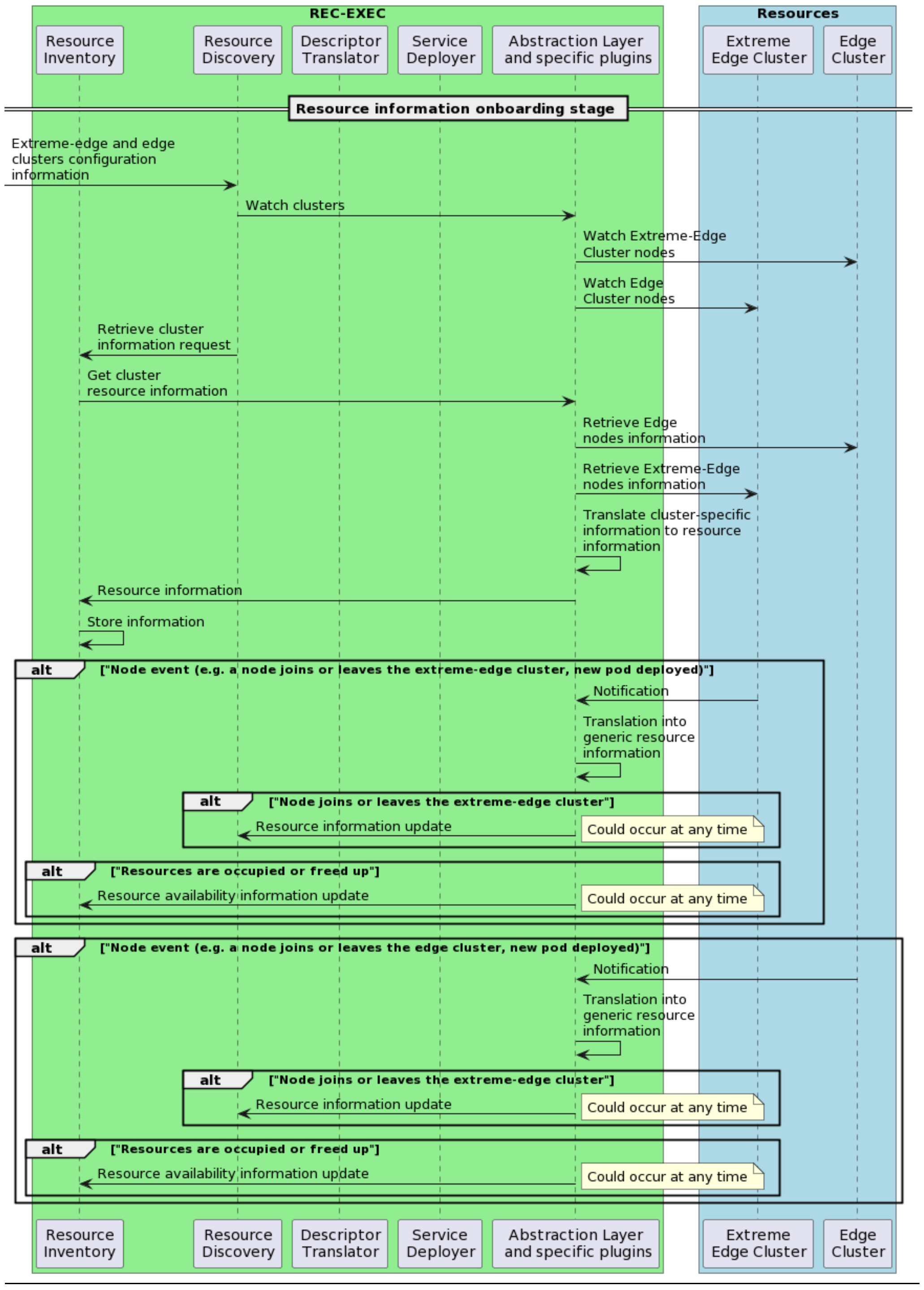

REC-EXEC
Resources
Resource Inventory
Resource Discovery
Descriptor Translator
Service Deployer
Abstraction Layer and specific plugins
Extreme Edge Cluster
Edge Cluster
Resource information onboarding stage
Extreme-edge and edge clusters configuration information
Watch clusters
Watch Extreme-Edge Cluster nodes
Watch Edge Cluster nodes
Retrieve cluster information request
Get cluster resource information
Retrieve Edge nodes information
Retrieve Extreme-Edge nodes information
Translate cluster-specific information to resource information
Resource information
Store information
alt ["Node event (e.g. a node joins or leaves the extreme-edge cluster, new pod deployed)"]
Notification
Translation into generic resource information
alt ["Node joins or leaves the extreme-edge cluster"]
Resource information update
Could occur at any time
alt ["Resources are occupied or freed up"]
Resource availability information update
Could occur at any time
alt ["Node event (e.g. a node joins or leaves the edge cluster, new pod deployed)"]
Notification
Translation into generic resource information
alt ["Node joins or leaves the extreme-edge cluster"]
Resource information update
Could occur at any time
alt ["Resources are occupied or freed up"]
Resource availability information update
Could occur at any time

Figure 5-7. Workflow for the discovery of extreme-edge nodes.

As can be seen, the workflow consists of the following steps:

1. The REC-EXEC is configured using the cluster configuration information (e.g., credentials, IP addresses, and so on) sent to the RD. In this way, the REC-EXEC knows where the resources (and thus the clusters) are located.
2. The RD, through the AL, performs a watch request to both extreme-edge and edge clusters and requests the RI information about the cluster's resources.
3. The AL translates the information about the nodes and their resources into more generic and platform-independent information stored in the RI.

4. At any given time, it is possible that some events can occur at the edge or extreme-edge cluster. For instance, a node can join\leave the clusters themselves or a pod is deployed. In this case, the RD is notified, and the RI is updated.
5. Thanks to the watch mechanism, this information is notified from the cluster itself to the specific plugin of the AL. Then, this event processed by the AL is notified to the RD, which updates the RI.
6. At this point, the extreme-edge and edge nodes and resource information are available in the REC-EXEC, within the RI.

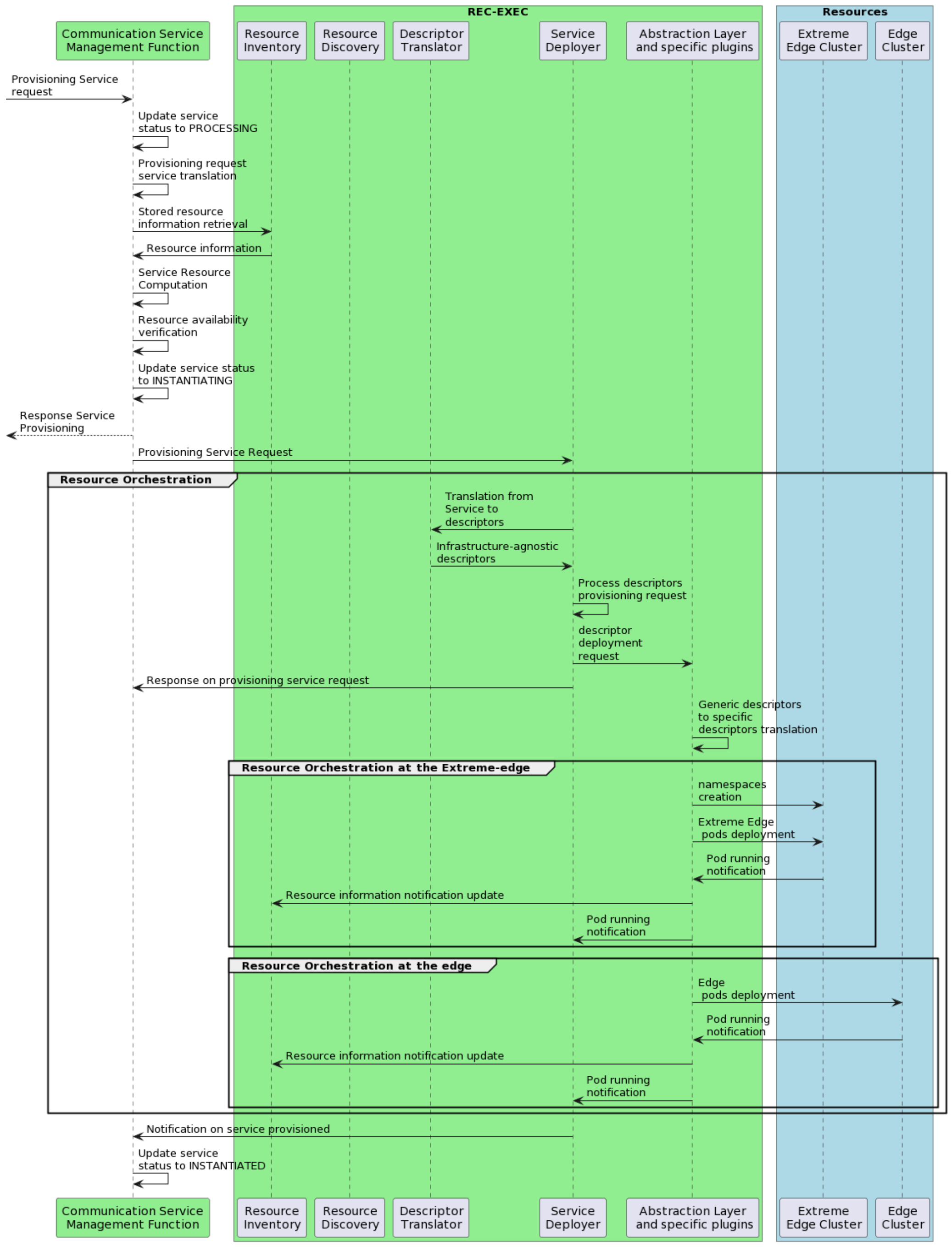

REC-EXEC
Resources
Communication Service Management Function
Resource Inventory
Resource Discovery
Descriptor Translator
Service Deployer
Abstraction Layer and specific plugins
Extreme Edge Cluster
Edge Cluster
Provisioning Service request
Update service status to PROCESSING
Provisioning request service translation
Stored resource information retrieval
Resource information
Service Resource Computation
Resource availability verification
Update service status to INSTANTIATING
Response Service Provisioning
Provisioning Service Request
Resource Orchestration
Translation from Service to descriptors
Infrastructure-agnostic descriptors
Process descriptors provisioning request
descriptor deployment request
Response on provisioning service request
Generic descriptors to specific descriptors translation
Resource Orchestration at the Extreme-edge
namespaces creation
Extreme Edge pods deployment
Pod running notification
Resource information notification update
Pod running notification
Resource Orchestration at the edge
Edge pods deployment
Pod running notification
Resource information notification update
Pod running notification
Notification on service provisioned
Update service status to INSTANTIATED

Figure 5-8. Over extreme-edge and edge continuum.

After this initial stage, the CSMF can receive requests for provisioning new services, which are allocated on the various clusters and nodes depending on the available resources.

The provisioning service time can vary, because mainly depends on the amount of resources to be allocated in the edge and extreme-edge. For this reason, asynchronous mechanisms between the service requestor and CSMF and between the CSMF and the REC-EXEC.

However, the workflow involving the CSMF itself and the REC-EXEC is realised, as depicted in Figure 5-8 is described below.

As can be seen, in this case, the workflow comprises the following steps:

1. The CSMF receives a request to provision a service. The request status is set to PROCESSING.
2. Then, the CSMF translates the provisioning request into a set of required resources needed for provisioning the service.
3. The CSMF, after retrieving the resource information from RI, computes the resource allocation for the service provisioning. If the number of available resources is enough, then the service is set in INSTANTIATING status.
4. The provisioning service requestor receives a notification about the service provisioning taken in charge because it is not known a priori the time to provision the service.
5. The CSMF sends the provisioning request to the SD.
6. The SD invokes the DT to translate the service provisioning request into a set of generic descriptors for the single target domains.
7. The SD notifies the CSMF about the fact that the provisioning request has been taken in charge, avoiding active waiting.
8. The descriptors deployment request sent by the SD to the AL is translated into a set of infrastructure-specific descriptors. In this particular case, the descriptors are related to the edge and extreme-edge namespace and pods deployment.
9. The namespaces and pods are deployed on the extreme-edge and edge by specific plugins of the AL.

Once the pods are running, the AL notifies:

- the RI about the resource used for the pods;
- the SD about the running status of the pods themselves;
- the SD that notifies the CSMF about the service provisioned.

At this point, the service has been successfully provisioned, deploying the pods to both edge and extreme-edge clusters. Moreover, the CSMF keeps the status of the service itself, as well as the REC-EXEC keeps the status of the different resources in terms of a node within the two clusters and is used for the whole service.

AI/ML-driven road traffic control functionality

Figure 5-9 depicts a high-level view of the flow diagram picturing the interactions among the main service components introduced in the previous Section 5.3.1.2. For the sake of simplicity, two of the components have been omitted: the Messages Broker component (RabbitMQ) and the SUMO Cloud Controller component, given that these two components do not play a relevant role in the main service logic (note that the Messages Broker is used just to communicate the SUMO Extreme-edge Controller, the AI Agent and Traffic Lights Control Logic components, while the SUMO Cloud Controller is just to boot-up the SUMO Server process and to get metrics from it). Besides, note also that one single instance of the SUMO Extreme-edge Controller and the Traffic Lights Control Logic is depicted, but it is important to recall that actually, they are four instances of these SW components running on the simulation (one for each crossroad/Raspberry Pi), and only one instance of the AI/ML component (although this one executes four RL Agents, as mentioned in Section 5.3.1.2). As can be seen, the SUMO Simulation starts by loading the required data for the simulation scenario in the SUMO Server (the two initial steps in the diagram). After that, the SUMO Server is left on a *listening* status, awaiting client connections.

The clients, in this case, are the four instances of the SUMO Extreme-edge Controller (i.e., the ones on each Raspberry Pi). Once connected, the clients request the initial traffic lights status for the crossroad they are operating onto (a random initial status generated by SUMO) and send that information to the *Traffic Lights Control Logic* component in order to synchronise the physical LEDs with the simulation status.

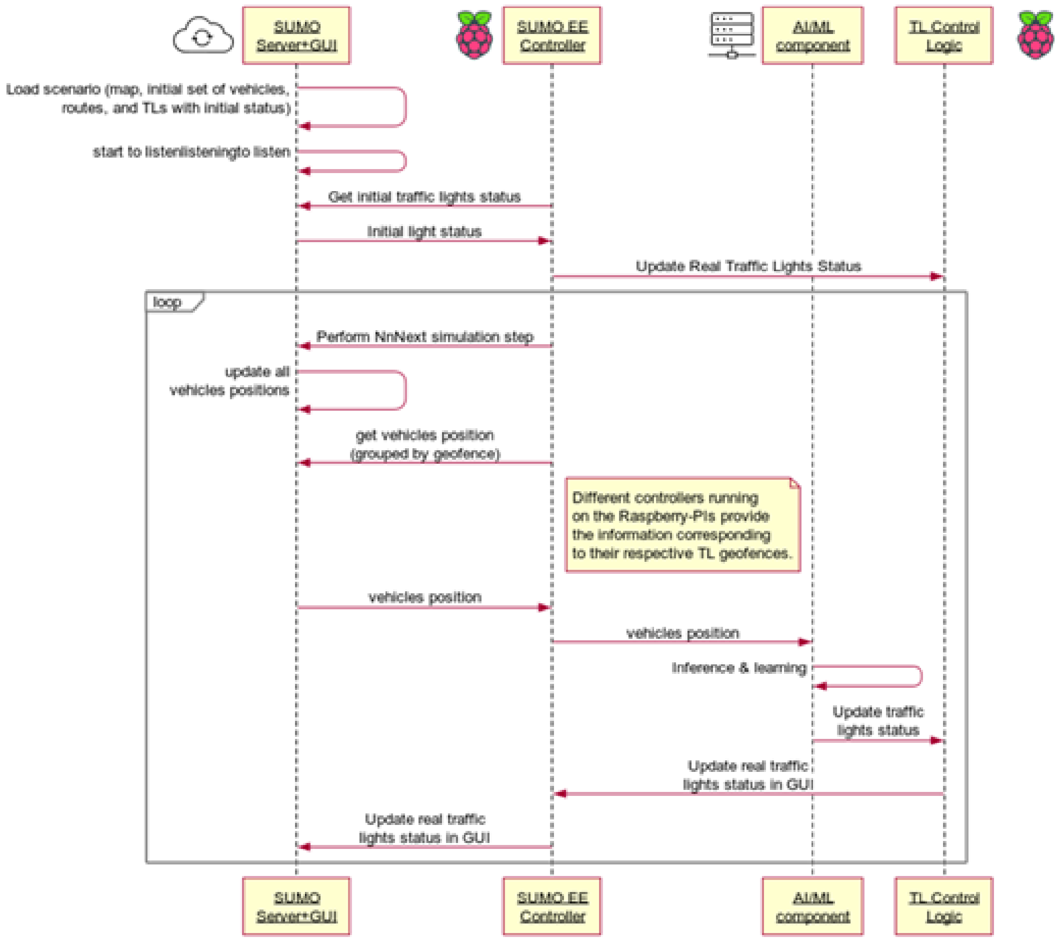

SUMO Server+GUI
SUMO EE Controller
AI/ML component
TL Control Logic
Load scenario (map, initial set of vehicles, routes, and TLs with initial status)
start to listenlisteningto listen
Get initial traffic lights status
Initial light status
Update Real Traffic Lights Status
loop
Perform NnNext simulation step
update all vehicles positions
get vehicles position (grouped by geofence)
Different controllers running on the Raspberry-PIs provide the information corresponding to their respective TL geofences.
vehicles position
vehicles position
Inference & learning
Update traffic lights status
Update real traffic lights status in GUI
Update real traffic lights status in GUI
SUMO Server+GUI
SUMO EE Controller
AI/ML component
TL Control Logic

Figure 5-9. Scenario 5.1 functional flow diagram.

Once this initialising sequence is finished, the simulation enters into its main execution loop, on which the following steps are executed:

1. A synchronisation signal is sent from each *SUMO Extreme-edge Controller* instance towards the *SUMO Server*. This is intended for all the components in the simulation to run properly in a synchronized way.
2. The *SUMO Server* updates all the vehicles' positions within the simulated scenario, e.g., moves vehicles to new coordinates, adds new vehicles, removes the vehicles that have reached their destination, etc. (this step is carried out only if all the components are properly synchronised – this is why the previous Step 1 is necessary).
3. Each *SUMO Extreme-edge Controller* instance collects from the SUMO Server the *status* information to the AI/ML component, i.e., the degree of occupancy of the monitored lane areas associated with each traffic light in each crossroad, and sends this information towards the *AI/ML* component.
4. Using this information, the RL Agents (one per crossroad) in the *AI/ML* component perform basically two tasks:
5. Using the data from the current iteration, they compute the *actions* to be done on their associated traffic lights (e.g., to change from red to green) based on the reinforcements received in previous iterations and send them to each *Traffic Lights Control Logic* component.
6. Measures the effect of the actions taken in the previous iteration considering the average road traffic speed in the simulation and, based on that, computes the *reward* and updates

the actions policies to be applied in future iterations, reinforcing those actions that, overall, improve the movement of the vehicle[5]. This in itself constitutes the *reinforcement learning* process of the AI Agents, which occurs continuously on each iteration[6].

7. Each *Traffic Lights Control Logic* instance applies its internal logic to the received data, process it, and changes the physical LEDs status if required. After that, it will send the new traffic lights status to its respective *SUMO Extreme-edge Controller* instance.
8. Each *SUMO Extreme-edge* instance will send the updated traffic lights status to the *SUMO Server* (which will display it on its associated GUI).

As shown in the diagram, this loop is executed repeatedly until the end of the SUMO simulation.

5.3.1.4 Deployment

Figure 5-10 shows the deployment diagram used for Scenario 5.1.

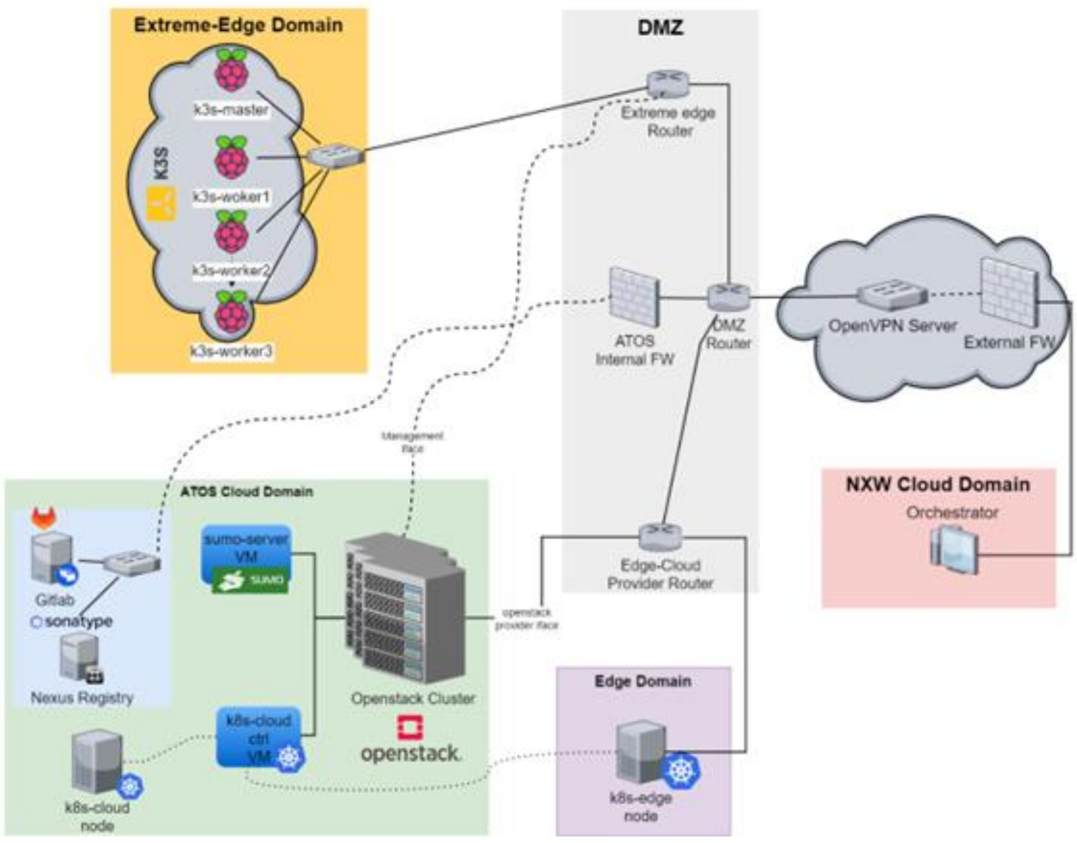


Figure 5-10. Scenario 5.1 deployment diagram.

As can be seen, the deployment includes different scopes:

[5] More information on the specific RL Agent implementation details can be found in Annex II.

[6] Please, bear in mind that in RL models there is not a clear separation between the training and the execution stages for the models, i.e., agents learn while they are actually interacting with the real environment on which they are integrated (the urban environment in this demo). In this specific case, in the initial stages of the simulation, i.e., while the model is still in its early stages of learning, the traffic lights just work in what we might call their "legacy mode", i.e., being controlled only by fixed time patterns, as it typically happens in the real-life scenarios (this legacy behaviour is determined by the "hard-wired" rules in the *Traffic Lights Control Logic* module). Later on, as the agent gains experience, the agent-generated actions predominate, so that the traffic control evolves into a more intelligent mode, better adapted to the traffic conditions generated in the simulation. This means that, although the learning takes place directly on the execution environment, in the worst case the behaviour of the traffic lights would always be timing-based (that of the "legacy mode"), so that even in the case where the RL agent might not work well yet, it would never cause harm, beyond generating the same kind of situations as those based on using regular timing patterns for the traffic lights control.

- the extreme-edge domain (orange block, top-left);
- the cloud domains (light green block at the bottom-left and red block at the bottom-right);
- the edge domain (purple block at the bottom);
- the DMZ block (light grey block in the middle of the figure).

The extreme-edge domain has been built using four Raspberry Pi cards (brand Broadcom, model BCM2711, with a 64-bit quad-core Cortex-A72 -ARM v8- @ 1.5GHz, and with 8GB RAM), bound together into the K3s cluster (the four of them are running Ubuntu v20.04.5). They are connected to each other using a Netgear GS308v3 network switch. The picture in Figure 5-11 shows the practical implementation of this small-scale extreme-edge domain deployed at Atos premises. Additionally, this extreme-edge domain also consists of a physical panel which mimics the simulated urban environment described before (see Figure 5-12 below). As it can be appreciated, this panel represents exactly the same urban environment as the simulated one, with the same streets, intersections, etc., but it also contains a realistic implementation of the traffic lights in the scenario by means of LED lamps, intended to emulate a realistic implementation of what would be a traffic lights set in a real urban scenario The idea behind this set-up is to try to represent an environment similar to what could be found in a real situation: an extreme-edge environment with low-power control devices activating real traffic lights and in communication with the edge and cloud nodes on which the M&O systems and the AI service components would run. Of course, the LED lamps in this panel are connected to the Raspberry Pi cards in Figure 5-11, where the traffic lights control logic modules are running.

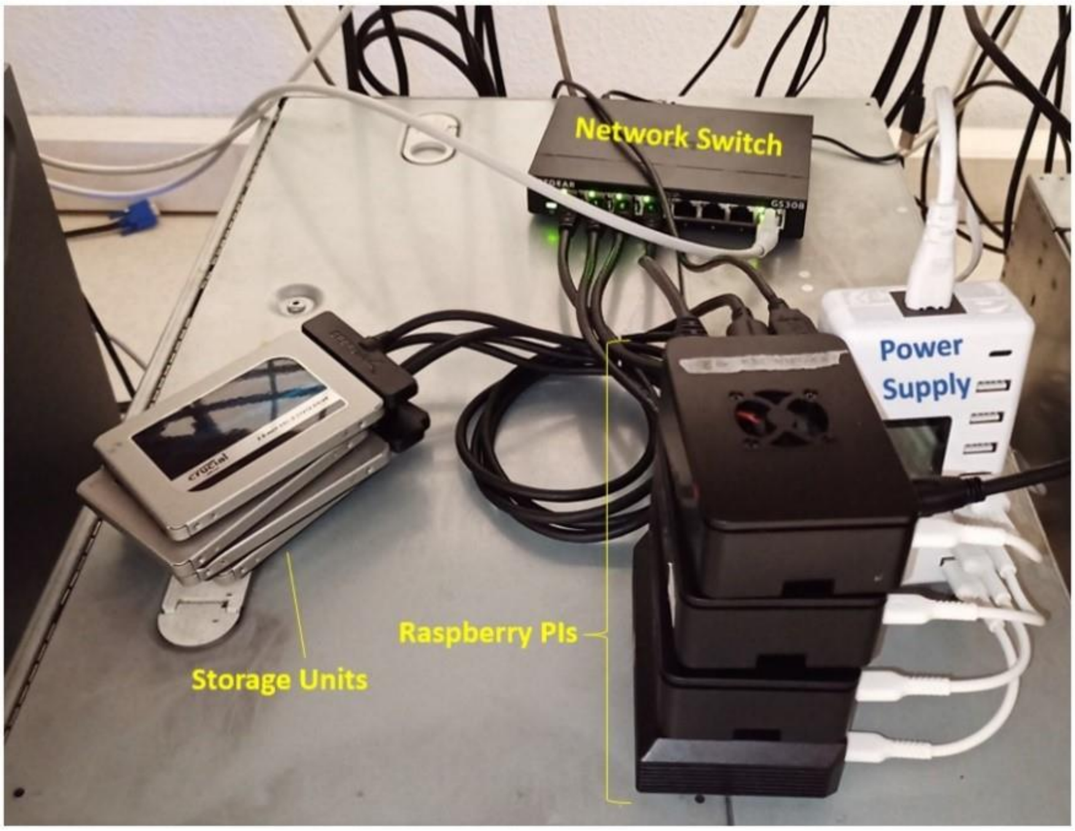


Figure 5-11. Scenario 5.1. Extreme-edge implementation.

Moving to the cloud domain, it has been implemented using physical resources from the two partners participating in this Scenario 5.1: Nextworks and ATOS. The Nextworks cloud part hosts the M&O system (the SEBASTIAN system – see Figure 5-4), while the Atos infrastructure is used to host the managed objects (those in Figure 5-4 as well). Both environments, in different geographical locations (in Italy and Spain), are connected through the Demilitarized Zone (DMZ) block represented in Figure 5-10.

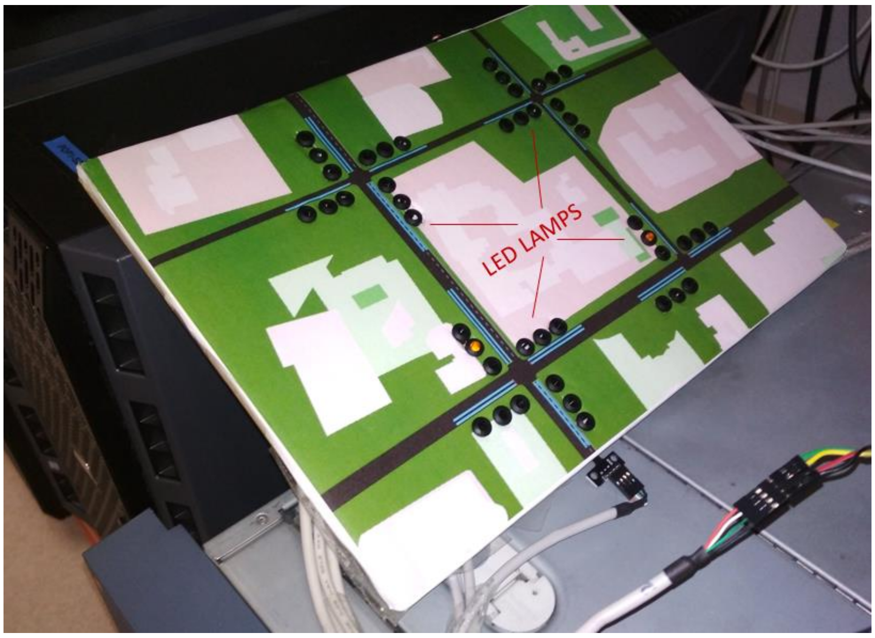


Figure 5-12. Scenario 5.1: Traffic Lights physical panel.

Specifically, the following physical resources have been used for each domain:

- On the Atos side (in Spain), the cloud has been implemented by means of a single, general-purpose Dell server (model PowerEdge T550) and an additional Intel NUC small-form computer (model NUC8i7HVK). The general-purpose server has been used to instantiate two VMs intended to run the K8s cloud controller and the SUMO server (see Figure 5-10). Table 5-1 summarizes the main features of these VMs. On the other hand, the cloud NUC is used to deploy all the required NFs (in the form of pods or CNFs) that might be needed in the cloud domain. Finally, within this domain, all the DevOps (i.e., Gitlab and Nexus container registry) repositories are allocated in servers shared across Atos.

Table 5-1. Virtual Machines in the Atos cloud domain.

Hostname	OS	Architecture	CPU (#)	RAM (GB)	Disk (GB)
K8s-cloud-ctrl	Ubuntu 22.04.1	x86_64	3	6	40
Sumo-server	Ubuntu 20.04.5	x86_64	4	8	40

- On the Nextworks side (in Italy), the orchestrator software stack relies on the internal Open Stack virtualized infrastructure composed of one controller node and two physical computers. On top of this virtualized infrastructure, it is installed the orchestrator software stack that can be deployed either as a single virtual machine or multiple virtual machines. Since the CSMF and REC-EXEC are logically separated, these have been installed and deployed on two different VMs whose hardware and OS requirements are available in Table 5-2. However, the deployment of CSMF and REC-EXEC can be performed using either Docker containers or K8s on those VMs. In the former case, the minimum required version is 20.10.13, while in the latter case, the minimum required is 1.21. As illustrated in Figure 5-14, end users are consuming a service running in edge servers connected via the radio access network. In conclusion, once the VMs where the CSMF and REC-EXEC are up and running (regardless of the compute nodes they rely on), a VPN is established towards the Atos testbed for making working the REC-EXEC communications towards the extreme-edge and edge clusters. Regarding the edge domain, it has been implemented using a NUC computer (same model as the one used for the cloud) that is used to execute the edge software components (those in Figure 5-4), which are always in the form of K8s pods. Following the description of the different components in Figure 5-10, the DMZ block is comprised of four network components. The “DMZ Router” acts as an isolating router that enables the generation of the Demilitarized Zone – DMZ block.

Table 5-2. Virtual Machines in the Nextworks cloud domain.

Hostname	OS	Architecture	CPU (#)	RAM (GB)	Disk (GB)
CSMF	Any compatible with minimum version of Docker or K8s	x86_64	4	4	40
REC-EXEC		x86_64	8	8	40

The “extreme-edge” and the “edge/cloud” routers provide two separate networks to the different components of each domain so that they can be logically and physically separated. The “Internal Firewall” network security component adds another security layer between the DMZ inner components and the external world.

<u>Demo presentation</u>

The demo is intended to be showcased including the following three elements (see Figure 5-13):

1. A main screen to show the real-time simulation execution (the SUMO Server GUI). This main screen is split into two (see Figure 5-13): the right-hand side showing the simulation, but without using the AI/ML algorithms developed for the demo (i.e., activating the traffic lights just using fixed time patterns). In this case, it is appreciated that several traffic jams will appear after a short period of time. At the same time, and on the left-hand side, a second window shows the same situation but controlled by the AI/ML service, resulting in an improved traffic density situation. This set-up allows the user to appreciate the advantage of the AI/ML approach at a glance by having the two simulations running in parallel.
2. A secondary screen that can be used for different purposes: as a console to show how the service can be deployed using the continuum orchestration function and also to show relevant service metrics.
3. The Raspberry Pis and the panel with the traffic lights are implemented with the real LED lamps (the one in Figure 5-12) to showcase that the simulation runs in sync with this realistic implementation of the extreme-edge domain.

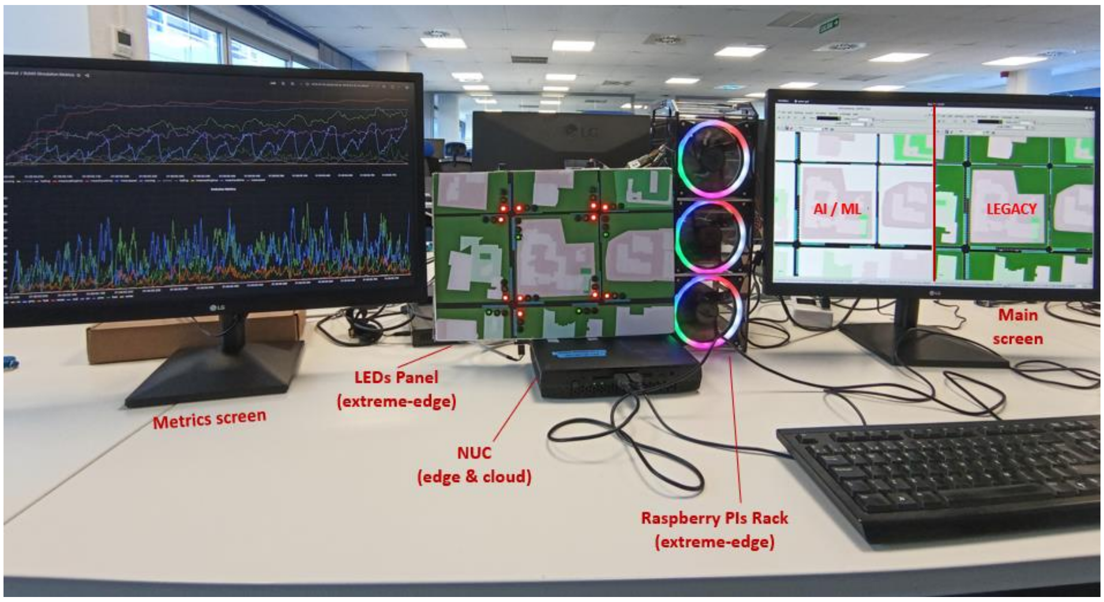


Figure 5-13. Elements for presenting Scenario 5.1.

In Annex III, some graphs are shown with the results from the demo execution using this set-up.

5.3.2 Scenario 5.2: Prediction-based URLLC service orchestration and optimization

5.3.2.1 Scenario description

Scenario 5.2 aims at demonstrating the ability of proper machine learning algorithms to anticipate the resource needs of the network and preemptively activate the related services so the application perceives no delay. In a nutshell, in contrast to reactive methods that may scale up/down the resources as the traffic load increases/decreases, Scenario 5.2 demonstrates the advantages of a proactive approach that does not involve the typical delays of reactive methods. This is particularly critical for deployments where resources are set in a deactivated or sleep state to support sustainability but require a non-negligible amount of time to be powered on. Such boot-up delays are highly harmful in the case of real-time services, such as URLLC services.

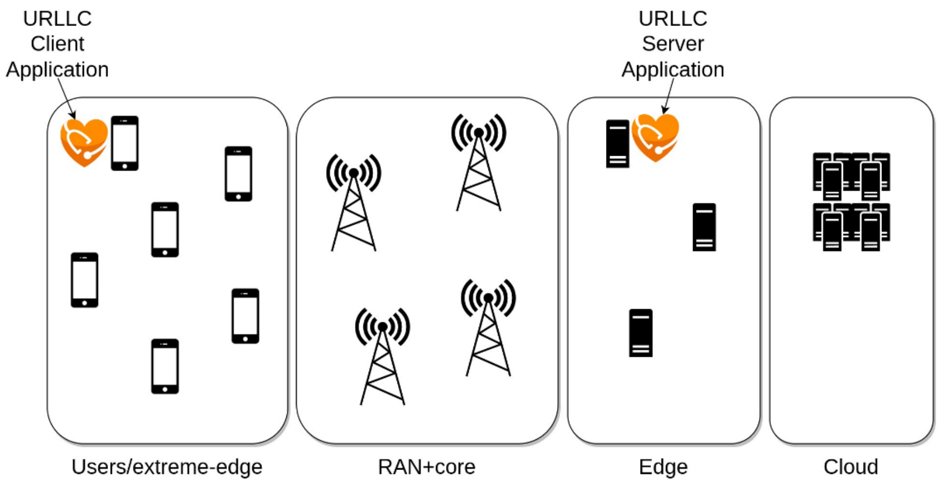


Figure 5-14. A high-level view of Scenario 5.2 configuration.

Scenario 5.2 complements Scenario 5.1 by focusing on a prediction-based orchestration of computing nodes, providing a URLLC service having stringent delay requirements. As illustrated in Figure 5-14, end users are consuming a service running in edge servers connected via the radio access network or running on an extreme-edge resource if available and if deemed suitable by the service orchestrator. It is assumed that the traffic requests follow a typical daily pattern.

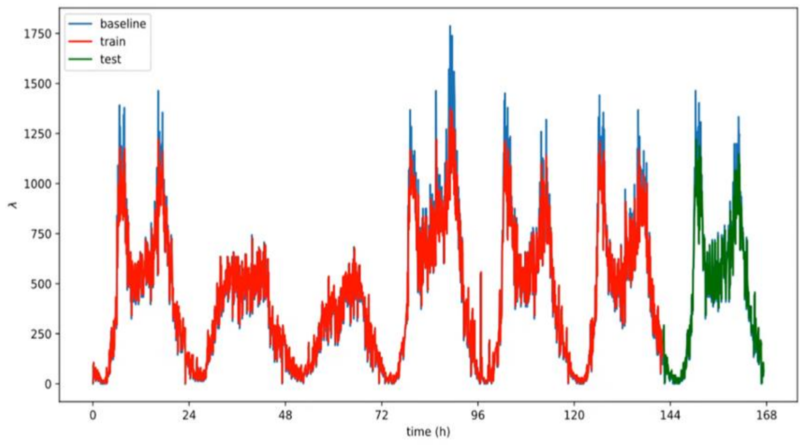


Figure 5-15. Exemplary traffic trace used in Scenario 5.2.

Figure 5-15 illustrates the vehicular traffic pattern over a week in an Italian city [VBM+21]. The different traffic peaks and valleys are noticeable and result in a drastic variation in traffic over time. Because of these variations, to support a sustainable service, it makes sense to implement a resource-on-demand policy, where the number of resources activated at the edge to provide the URLLC service (e.g., real-time video processing) matches the demand at a given point in time.

One challenge with this approach is that the time to boot up a machine is longer than the inter-arrival time of requests, and therefore reaction methods are, in general, not adequate to support a timely matching of resources to the required capacity. In contrast to reactive methods, this demo illustrates the advantages of using a prediction algorithm based on machine learning. More specifically, it illustrates how a Long Short-Term Memory (LSTM) network [LST] can predict the traffic during a given future time window (in green, in the figure above), and therefore providing the required anticipation to the orchestration mechanism to activate the required resources and provision the service with zero-perceived disruption.

5.3.2.2 Software Components

Scenario 5.2 includes three main software components, as Figure 5-16 shows. The first component is the URLLC application, which comprises a client and a server. A second component is the Simu5G network emulator [NSS+20], which is responsible for emulating the communication environment and the edge-cloud computing environment. Finally, a third component is an intelligent orchestrator. In the following, we provide the details and relations of each component.

URLLC Application

The scenario assumes an application with stringent delay and delivery guarantees, such as those required by vehicular traffic (e.g., teleoperated driving [5GA21]). It should be noted, though, that the use of this type of application would be unpractical, and there is no open-source software available. Furthermore, in order to measure and keep track of the experienced delay, it becomes advisable to rely on a synthetic traffic generator to assess the performance of the system. Because of this, the iPerf traffic generator [IPE] is considered.

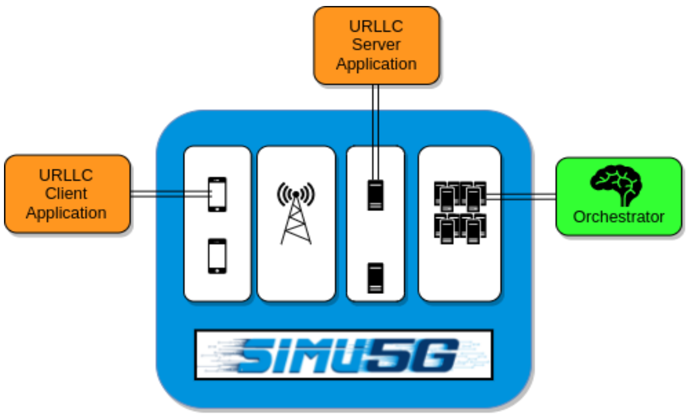


Figure 5-16. Main software elements composing Scenario 5.2.

Network Emulator

The software component used to emulate the network is Simu5G, which is the evolution of the well-known SimuLTE [VIR+16], extending 4G capabilities with 5G capabilities at both radio access and core network side. Simu5G is an event-driven system-level simulator building upon models from the INET library [INE], which allows one to simulate a complete TCP/IP-based network stack and supports end-to-end communications among applications. Simu5G models the data plane of both the core and the radio access networks. As far as the Core network is concerned, it allows users to instantiate a User Plane Function (UPF) or Packet Data Network GateWay (PGW) and an arbitrary topology, where forwarding occurs using the GPRS Tunnelling Protocol (GTP). As far as radio access is concerned, it allows one to instantiate gNBs and UEs, which interact using a model of the New Radio (NR) protocol stack. The gNBs can be connected to the Core network directly in the so-called standalone deployment. Alternatively, a gNB can operate in an E-UTRA/NR Dual Connectivity (ENDC) deployment, wherein LTE and 5G coexist. The gNB are connected through the X2 interface, and all 5G New Radio (NR) traffic traverses the eNB first. UEs and gNBs are compound OMNeT++ modules. UEs have all the protocol stack until the application layer, whereas gNBs only have Layer 3 functionalities. Both include an NR Network Interface Card (NIC), which models the NR protocol stack. Packet transmission entails top-down traversal of the NR protocol stack, with messages exchanged by neighbouring modules. Conversely, packet reception entails bottom-up traversal. Note that OMNeT++ messages are events: the price to pay for complete modelling of the layers within the NR protocol stack is that the transmission of a single IP packet via the NR interface requires Simu5G to handle a sizable number of events in the order or few tens, among inter-layer communication, fragmentation/reassembly, timers, ACK/NACK sending, etc

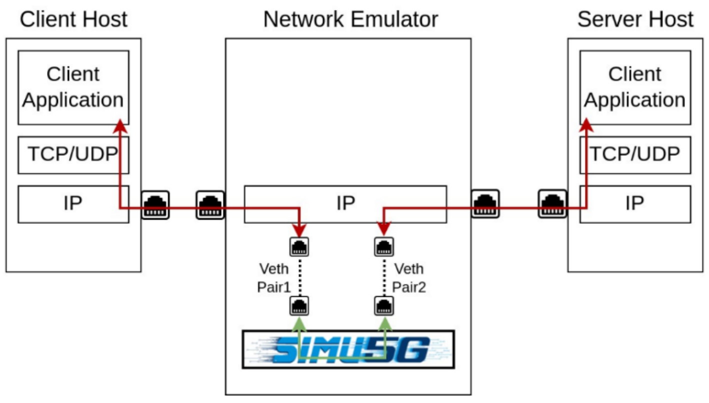


Figure 5-17. URLLC traffic flow in the Simu5G-based emulated network.

The gNB are connected through the X2 interface, and all 5G New Radio (NR) traffic traverses the eNB first. UEs and gNBs are compound OMNeT++ modules. UEs have all the protocol stack until the application layer, whereas gNBs only have Layer 3 functionalities. Both include an NR Network Interface Card (NIC), which models the NR protocol stack. Packet transmission entails top-down traversal of the NR protocol stack, with messages exchanged by neighbouring modules. Conversely, packet reception entails bottom-up traversal. Note that OMNeT++ messages are events: the price to pay for complete modelling of the layers within the NR protocol stack is that the transmission of a single IP packet via the NR interface requires Simu5G to handle a sizable number of events in the order or few tens, among inter-layer communication, fragmentation/reassembly, timers, ACK/NACK sending, etc. From a physical layer standpoint, Simu5G models the effects of propagation on the wireless channel at the receiver without modelling symbol transmission and constellations. When a sender sends a MAC Protocol Data Unit (PDU) to a receiver, the PHY modules of the two entities exchange an OMNeT++ message, whose propagation delay is set to the duration of one NR time slot.

Within the scope of Demo #5, Simu5G is used as the network transport, having application endpoints exchanging packets through it in real-time. In the literature (e.g., [CAR+03], [MAH+04]), such an approach is referred to as emulation since packets exchanged by real applications with the simulator perceive the same impairments (e.g., delay and losses) as if they were running on the real network. This is useful to test and showcase the real-time performance of an application, e.g., when closed-loop sensing and control applications are to be tested. These applications can be, for instance, the two counterparts of a MEC-based URLLC application, one running on a 5G UE in mobility and the other on a MEC host connected to the 5G infrastructure. This allows us to test the performance of our scenario on a 5G network under controlled conditions (e.g., as for load, channel quality, mobility, etc.) in a preproduction environment so as to obtain confidence regarding their performance. Within the scope of Demo #5, one real application generates real network traffic to be sent through the Simu5G network emulation. The URLLC application includes the client and server sides, running on two separated hosts, as Figure 5-17 shows. The network traffic between the client application (Host B) and a server application (Host C) flows through a Simu5G instance (Host A). The client application transmits data packets over a TCP socket to the server application. These processes are unaware of the presence of Simu5G. The OS on host A takes care of forwarding their packets through Virtual Ethernet (*veth*) interfaces, as depicted in Figure 5-17 when the sender transmits data via a UDP socket by specifying the IP address of *veth2* and the port number the receiver is listening to, while the routing table of the host is configured to reroute packets destined to *veth2* through *veth1* and vice versa, the OS forwards them to the Virtual Ethernet interfaces depicted in Figure 5-17. The packets are forwarded through the 5G RAN and reach a MEC application on an emulated MEC host. Finally, the packets exit the emulated network through *veth2* and are forwarded to the server application.

Orchestration Interface

The Orchestration Engine allows the orchestrator to interact with the system emulated within Simu5G. On the one hand, it allows the enforcement of orchestration decisions on the emulated resources. This includes:

- the activation/deactivation of edge nodes. Although a node can be activated instantaneously, activating it requires a configurable amount of time;
- the offload of the servicing application between edge and extreme-edge nodes and the consequent redirect of client requests to the proper service;
- the (re)balancing of existing services among the active edge nodes. This operation is performed in a simulated manner in the scope of this demo scenario; however, the procedures and methods for migrating running services between edge servers have been defined and validated in [BPV+22].

On the other hand, the orchestration engine retrieves information on network status and feeds it to the intelligent orchestrator. The information collected includes the current load of the network in terms of the active application, as well as the location and status (reachable/unreachable) of the existing extreme-edge resources. The orchestration engine is executed periodically within the emulated network. At each execution, it performs the following operations:

- it retrieves the current number of services in the system;
- it retrieves the status of the extreme-edge resources;
- it performs a remote call to the intelligent orchestrator, passing the collected information;
- it retrieves and parses the orchestration decision;
- it enforces the actions stated in the orchestration decision, possibly activating and/or deactivating edge nodes whenever needed.

The Orchestration interface has been developed specifically for the purpose of Scenario 5.2.

Scenario Visualization

During the execution of the scenario, the status of its components and their statistics are continuously monitored and visualized through a GUI.

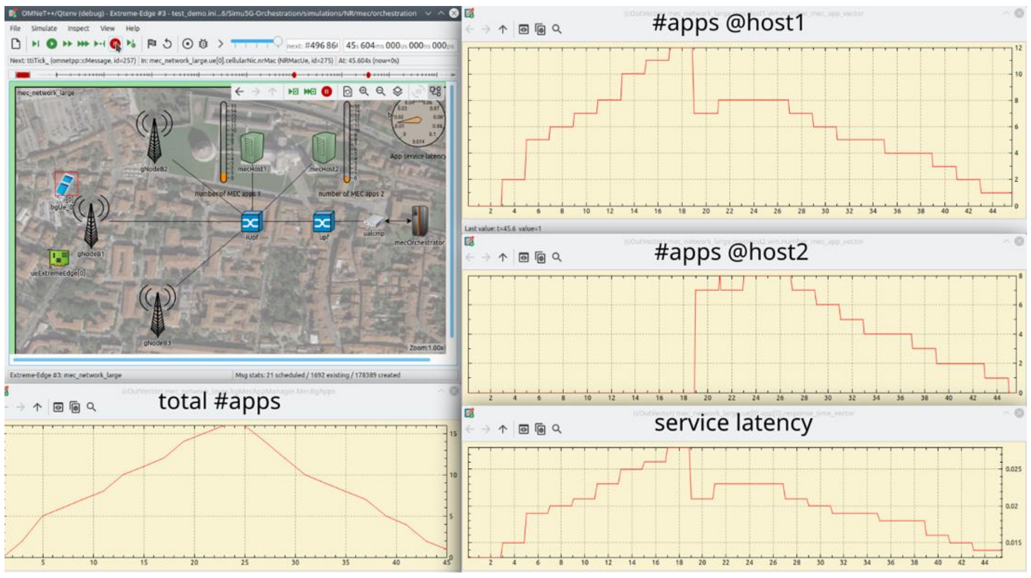


Figure 5-18. Real-time scenario visualization using a custom GUI.

Figure 5-18 provides an exemplary instance of such an implementation, which includes (i) the performance over time in terms of Service Time (QoS) of a monitored UE running the URLCC application client, (ii) the performance over time in terms of processing time of a monitored MEC Host, iii) the current deployment of network and computing nodes (e.g., gNBs, edge hosts, etc.) and the mobility of UEs, iv) aggregated statistics of traffic and loads.

Intelligent Orchestrator

Intelligent orchestration builds on two main functionalities. On the one hand, the ability to predict the traffic demand, i.e., a Prediction Module, and on the other hand, the Orchestration Intelligence. The traffic prediction takes as input the traffic and keeps the history of past traffic data to generate an AI model. The Orchestration intelligence is fed with the current traffic load and, based on the Prediction, takes an orchestration decision.

The intelligent orchestration entity is implemented as a client-server service. The server is logically executed on a remote node (e.g., in the cloud) which runs both the actual prediction and the orchestration logic and notifies the decisions to the client. The latter is logically executed at the edge and interfaces with the network emulator that finally enforces the orchestration decisions.

This Intelligent Orchestrator has been developed specifically for the purpose of Scenario 5.2.

5.3.2.3 Functional Behaviour

The demo is composed of three main functional blocks:

- Simu5G: which simulates the complete data plane of a 5G network. It relies on an "input trace" file, which consists of a set of {<time>, <no. of users>}tuples that determine the number of users present in the system at a given time. The simulator periodically provides as output to the orchestration intelligence the current status of the system (number of users, number of active servers, current time) and reads as input the decision from the orchestration intelligence (activate a server, deactivate a server or re-route the traffic from the target app through the extreme-edge).
- Intelligent Orchestrator: a module that is in charge of taking as input the current status of the system and the output of the prediction algorithm (discussed next) and producing as output the decision to be made at that point in time. The decision could be to activate a new resource, deactivate it, or re-route traffic via the extreme-edge.
- Prediction module: this module takes as input the "input trace" file and produces as output the predicted number of users over time. This prediction can be based on a simple Exponentially Weighted Moving Average (EWMA), which serves as a benchmark, or it can be based on a tong short-term memory artificial neural network. For the latter, in addition to the specifics of the neural network, it can be configured with a given "memory" window (i.e., the amount of prior data to be used) and another "prediction" window (i.e., how far ahead the algorithm should try to predict).

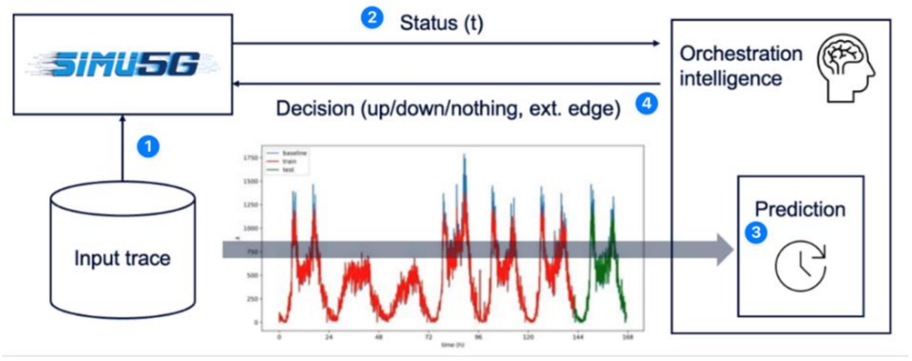

SIMU5G
2 Status (t)
Decision (up/down/nothing, ext. edge) 4
1
Input trace
Orchestration intelligence
Prediction
3
baseline
train
test

Figure 5-19. Functional blocks of Scenario 5.2.

In Figure 5-19, the different functional modules and their relationships are illustrated. The main interactions between modules are as follows:

1) The input trace (bottom left) feeds the simulator (top left) with real-life data about the number of vehicles.
2) The simulator provides the current status of the system to the orchestration intelligence (top right)
3) The orchestration intelligence feeds the information from the status of the system to the prediction algorithm (i.e., it "learns“ the input trace via the simulator, as denoted by the bottom arrow).
4) The orchestration intelligence, based on its information, provides the simulator with the orchestration decision to take.

5.3.2.4 Deployment

The deployment of Scenario 5.2 is shown in Figure 5-20. Simu5G is executed on a dedicated node and shows the status of the execution through a dedicated dashboard. A URLLC application, modelled in the form of an iPerf instance, is executed on a real device, a laptop in this case, and its traffic is routed into Simu5G, wherein it enters the network from the perspective of a simulated user.

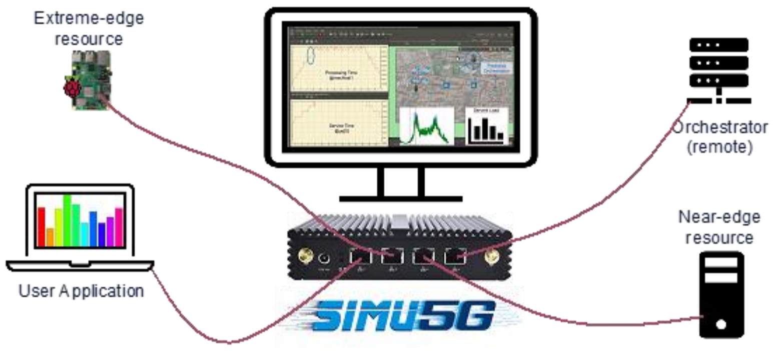


Figure 5-20. Deployment of Scenario 5.2 – architecture.

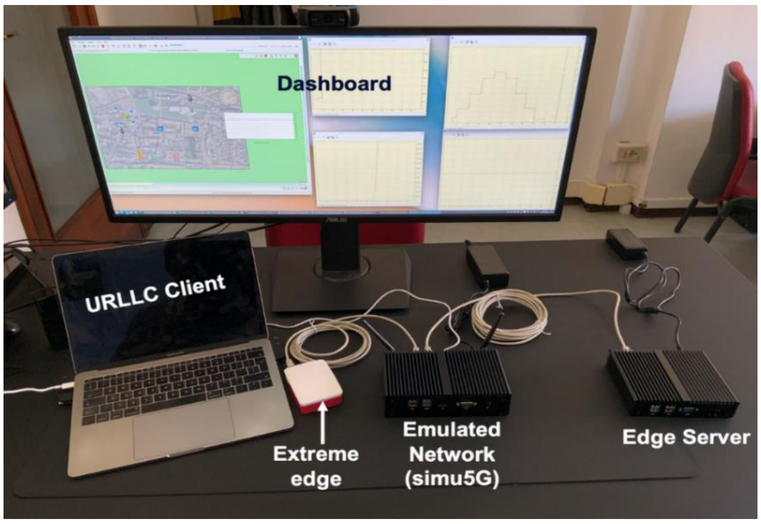


Figure 5-21. Deployment of Scenario 5.2 – real-life testbed.

The traffic travels through the emulated network and is affected by system delays. Traffic reaches the intended emulated endpoint (either an extreme-edge or a near-edge device) and exits the emulated network. The traffic is finally routed to the propped real endpoint: edge resource, implemented with a Raspberry Pi. Besides all of this, on the control plane, an orchestration application runs on a remote PC (which is logically in the cloud) and orchestrates the emulated resources, activating/deactivating edge nodes and/or offloading traffic to and from the extreme-edge resources. According to the description above, Scenario 5.2 can be deployed flexibly using different hardware devices. Figure 5-21 shows a real-life testbed which includes the devices, which are also described in Table 5-3.

Table 5-3. Description of the nodes used in Scenario 5.2

Name	Device	Role	OS	CPUs	RAM (GB)	Disk (GB)	Notes
SIMU5G	Qotom MiniPC – high performance	Emulated network using Simu5G	Ubuntu 20.04	Intel i7	8	128	Simu5G ver1.2
Edge Resource	Qotom MiniPC – low Performance	Edge server	Ubuntu 18.04	Intel Celeron	8	58	
User Application	Laptop MacBook Pro	User Device	macOS Big Sur	Intel i5	8	250	
Extreme-edge resource	Raspberry Pi	Extreme-edge resource	Raspian OS	Cortex-A72	4	64	
Orchestrator (remote)	Remote Server	Orchestration intelligence	Ubuntu 20.04	Intel i7	16	1000	-

5.3.3 Scenario 5.3: Reactive security for the edge

5.3.3.1 Scenario Description

The use cases developed in the previous Scenarios 5.1 and 5.2 regarding road traffic related applications, involve resources and services deployed over the extreme-edge. In these specific scenarios, the traffic lights are controlled by services hosted in Raspberry Pis, which, in the real world, would either be hosted within the traffic light itself or in a separated equipment in the direct vicinity of the controlled traffic lights. In both cases, a critical service is hosted on small, isolated spots of resources. By nature, those isolated resources could be cut off from the central clouds, either by accident or due to an attack. An attacker can also physically access those resources that cannot be closely guarded. At the same time, the service provided here is highly critical. If an attacker manages to disrupt the service, traffic may get slowed. Worst, if an attacker manages to take control of the service, it could cause deadly accidents within seconds. A solution is needed to protect those remote, extreme-edge services.

Scenario 5.3 precisely aims to demonstrate the ability of the proposed M&O architecture to efficiently handle cyber-security threats against a vulnerable application deployed at the extreme-edge. The addressed vulnerability in the demo scenario is *log4shell*, also known as CVE-2021-44228 [CVE-L4J], which is a zero-day vulnerability exploited by arbitrary code execution and affecting the Java utility *Log4j* [L4J]. To detect and remediate an attack exploiting this vulnerability, we propose hierarchical security management of two layers designed to accommodate the scarce resources available on the extreme-edge while ensuring the required security level.

Although standards establish a set of security measures to protect telecommunication networks, those networks still face risks of attacks. Those risks may come from unidentified/unknown threats and vulnerabilities, from trade-offs between costs of protection versus risk, or from poorly implemented standard security measures. To enhance security, it is then necessary to add to these preventive measures a reactive line of defence, to monitor continuously both the network itself and its users and detect any security event. 6G networks are envisaged as very complex systems to manage as a result of the distributed topology, which tends to include more edge devices, heterogeneous virtualization technologies of resources and functions, and sophisticated technologies that require significant expertise to master. Therefore, network service management, including security service management, needs to be assisted by means of automation. Such automation is usually implemented by several autonomic closed loops monitoring and acting upon the system. In this context, two problems arise for the security management of 6G networks:

- Resource locality. In 6G, many services may have to be located over extreme-edge resources, which are typically scarce and located far from a large central data centre. However, those services need security as well. Locating the associated security closed loop in the central cloud may induce a number of issues: it could be temporarily disconnected from the extreme-edge location, leaving it without security, it would induce additional bandwidth consumption to bring raw monitoring data from the edge to the central cloud, and it would induce additional latency due to the travel time. On the other hand, locating the security closed loop that has sophisticated activities on the extreme-edge premises may have a severe impact on the resources available, as security may require complex autonomous algorithms or/and large signature databases to analyse and respond efficiently to potential threats.

- Reaction time. Although the security processes rely on a closed loop, some actions of the loop (analysis, remediation plan creation) may take some time, enough for the attacker to inflict significant damage to the system. In some cases, the response would even involve requesting actions or authorizations from human actors, which considerably increases the response time.

To tackle these two issues, this Scenario proposes a layered security architecture aligned with the M&O architectural design provided in the previous Deliverable D6.2. Local security orchestrators are installed on the same premises as the assets they protect, including extreme-edge resources, while central security orchestrators are installed in a remote location with access to a vast pool of resources. Both types of orchestrators contain autonomic closed loops for threat detection and remediation. The local security orchestrators are intended to be lightweight to minimize their footprint on scarce local resources. To reach this goal, they would leverage the relationship with central security orchestrators as much as possible to coordinate activities, thus avoiding performing complex and computationally intensive tasks. The local security orchestrators take rapid and simple mitigation actions to quickly stop an ongoing attack, while the central ones take more complex actions to eradicate the attack and provide long-term protection to the system. In the context of AI used for security, central security orchestrators would typically centralize training data and train AI/ML models and would send the trained models to local security orchestrators to enhance their performance with little computational effort on their side. As a result, the local security orchestrators are less resource-consuming than the central one while still being in a position to provide an acceptable level of security for the assets under its control, even in the event of a temporary loss of connection to the central orchestrator. This layered security concept can further be extended to match the layered nature of the 6G network: each sub-slice and slice may have its own security orchestrator, with additional ones at the inter-slice level. In the specific perimeter of this scenario, the approach consists of a two-layer implementation, including a local and a central security orchestrator.

To demonstrate the efficiency of this solution, Scenario 5.3 focuses on one specific attack: *log4shell*. This attack takes advantage of a vulnerability in the *log4j* library, a popular Java logging library, to effectively give access to a shell in the target. Due to the widespread use of *log4j*, and the high impact of the attack, *log4shell* obtained a CVSS score of 10 out of 10 [L4J]. To deal with this attack, we propose a two stages process, represented in Figure 5-22. First, the attack is detected and mitigated locally. The mitigation is very fast and simple: it consists in blocking the source of the attack. The detection is propagated to the central security orchestrator, which takes more complex eradication actions: if allowed, it automatically patches the vulnerable application to a more recent version of *log4j*, immune to the attack. Else, it raises an alarm to request administrators to perform this patch or a full upgrade. Finally, the central security orchestrator can locate other applications within its management domain with similar vulnerabilities and apply the same eradication action to prevent any attack attempt. Due to the high versatility of the proposed management and orchestration architecture [HEX22-D62], the security service management enables dynamic communication across security functions and even M&O layers. Such security functions aim at providing accountability for security risk mitigation, forensic analysis and threat prevention, for instance, the aforementioned local and central security loops. In this vein, a security function may support to or consume from other security functions to ameliorate the overall 6G network services.

Complementary to the previous, when it comes to security management, Hexa-X has introduced the concept of Level of Trust Assessment Function (LoTAF) [HEX22-D14], which enables assessing the security and privacy aspects of a network service in a particular application environment. LoTAF is one of the security functions belonging to the Service Layer of the M&O architecture design [HEX22-D62], so it is totally aligned with the hierarchical security management architecture mentioned above. Thus, LoTAF may be considered as an add-on or supplementary solution (i.e., another security function developed for this Scenario 5.3), together with the local and central security orchestrators, to complement the security of future 6G network services. Therefore, LoTAF does not disrupt or modify the functionality of other security functions, but it mainly consumes information generated by other security functions to enhance the user's experience. Concretely, one of the principal actions under LoTAF is to verify whether new cyber security threats, appearing at the runtime stage, may compromise a set of security and privacy requirements previously requested by end-users during Stage 1 of Level of Trust (LoT) [HEX22-D14]. In this regard, both the local and the central security orchestrators may feed to the LoTAF with real-time information related to cyber security threats, *log4shell* attack in this Scenario 5.3, so as to update the Level of Trust (LoT) based on decisions and the kind of applied countermeasures (containment or eradication plans).

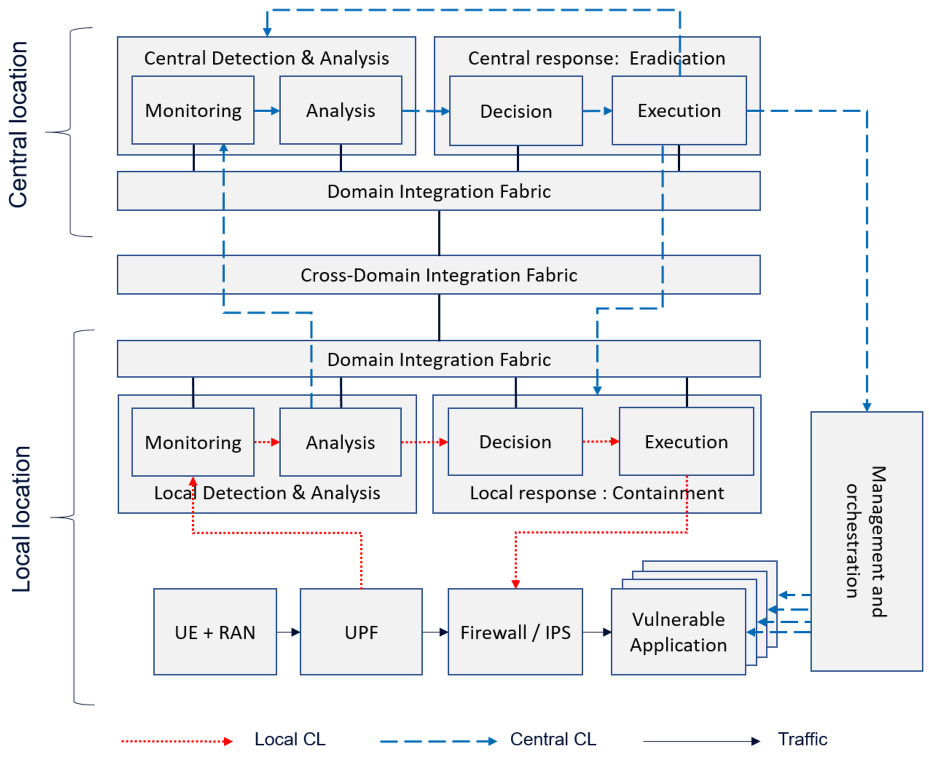

Central location
Central Detection & Analysis
Monitoring
Analysis
Central response: Eradication
Decision
Execution
Domain Integration Fabric
Cross-Domain Integration Fabric
Domain Integration Fabric
Monitoring
Analysis
Local Detection & Analysis
Decision
Execution
Local response : Containment
Local location
UE + RAN
UPF
Firewall / IPS
Vulnerable Application
Management and orchestration
Local CL
Central CL
Traffic

Figure 5-22. Local and central loops of Scenario 5.3.

It is worth mentioning that LoTAF is made up of two phases [HEX22-D14]. Stage 1 intends to assess the achievable LoT; in contrast, Stage 2 is centred on evaluating the achieved LoT once a network service is being leveraged. Owing to the fact that Scenario 5.3 is principally focused on monitoring, analysing and handling *log4shell* threat, only the LoTAF Stage 2 is going to be showcased since the main objective of Stage 1 is to discover from available network services, which one may ensure a set of an end user's security and privacy requirements.

5.3.3.2 Software Components

The software components used in this scenario are all represented in Figure 5-23 in green boxes:

- The UERANSIM [UER] (bottom-left in Figure 5-23) is a tool provided by Free5GC that allows a simulation of both a UE and a RAN. As an output, this component produces the equivalent of a RAN output, which includes both NAS and AS messages.
- VPP [VPP] (bottom middle in Figure 5-23) is an open-source software packet processor based on a Cisco commercial product. VPP presents very high performances, suitable to

handle the data plane traffic. The Control Plane (not represented in Figure 5-23) is made of a custom 5G Control Plane.

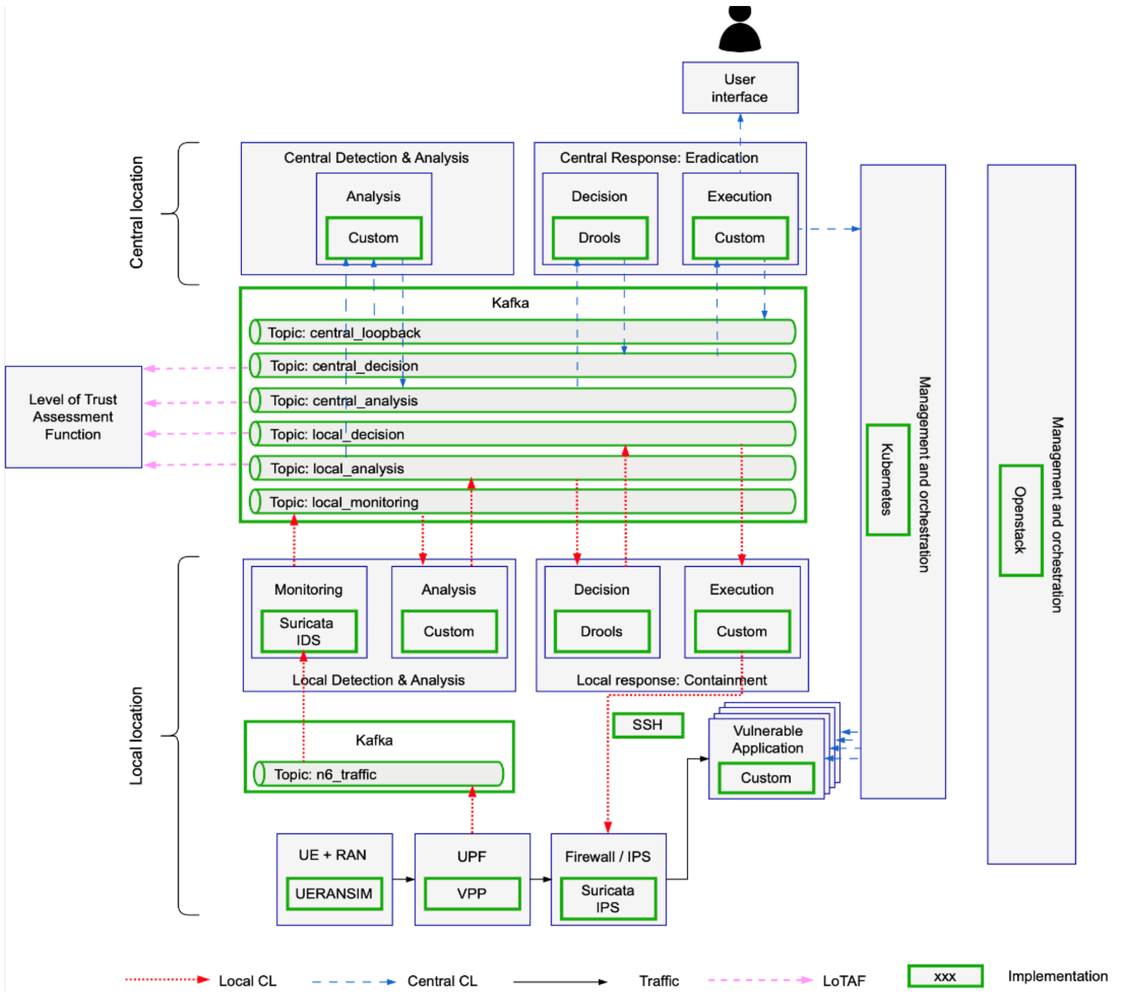


Figure 5-23. Software components used in Scenario 5.3.

- Suricata [SUR] (bottom centre and middle left in Figure 5-23) is open-source software that can be used either as an Intrusion Prevention System (IPS) or an Intrusion Detection System (IDS). The system uses both modes: the IDS as a monitoring tool and the IPS as a firewall. Used as a firewall, Suricata offers the advantage of being able to analyse traffic up to the application layer. Regarding monitoring, Suricata has been chosen as it comes with adapted rules to detect the *log4shell* attack. Additionally, the detection is very fast compared to another well-known monitoring tool, Zeek [ZEE], which works with batches of packets. However, it should be noted that the proposed solution allows to easily plug other monitoring systems in parallel with Suricata, if needed. The vulnerable application is an application built on purpose to demonstrate the *log4shell* attack in a K8s context. It has been taken from [VIC].
- Apache Kafka [KAF] (middle of Figure 5-23**)** is then used to exchange the necessary messages between the user plane and the security control loops and between the security control loops components. This platform has several advantages for our architecture. First of all, it can receive and distribute events with high performances and built-in scalability, which is important when it comes to handling network traffic. Secondly, the messages can be isolated in a different topic, which allows the creation of a pipeline where each module consumes from one topic and produces to another.
- The decision engines in both local and central loops are based on Drools [DRO], an open-source Business Rules Management System (BRMS) written in Java. This tool relies on a set of rules - a knowledge base to determine an action based on a given input. The local

execution module is custom to this scenario and mainly consists of a Kafka consumer coupled with an SSH client.

- Analysis modules in both local (bottom left in Figure 5-23) and central (top right of Figure 5-23) loops are both custom modules. In this scenario, their role is limited to applying a correct format to the alert since the alert is fully qualified by Suricata IDS and does not require further analysis.
- The local execution module (middle right in Figure 5-23) is a custom module. It is composed of a Kafka consumer and an SSH client connected to Suricata IPS VM. This module can push new rules into Suricata IPS and perform live-reload.
- The global execution module (top right of Figure 5-23) is a custom Kafka consumer to receive messages from the decision engine, a K8s client to search for the pod targeted by the attack, a library to look for other vulnerable applications (pods) and an SSH client. The SSH client is used to automatically trigger the *log4shell* hot patch solution proposed by the AWS Coretto team [COR]. The library used to scan for vulnerabilities is the one provided by Trivy [TRI]. Trivy is a scanner tool that can, among other things, detect vulnerabilities on container images.

This system is open to further evolution: while, in this demo, we have only one component to fulfil each function (one component for monitoring and one component for analytics, and so on), Kafka allows for easy introduction of other components in parallel with existing ones, as several components can consume/produce in the same topics. Hence, if further attack scenarios require a specific component, for example, an AI-based analytic engine, this tool can easily be plugged into the pipeline. Similarly, other components that are not directly part of the cybersecurity loop can also consume the topics for their own purposes. In this scenario, it is the case for the LoTAF module.

5.3.3.3 Functional Behaviour

As detailed in Section 5.3.3.1, the objective of this scenario is to demonstrate the ability of the proposed architecture to automatically detect, contain and eradicate cyber-attacks, specifically here, the *log4shell* attack. Furthermore, the outputs generated during the detection, containment and eradication steps are going to be used for adjusting the LoT of a network service, in this case, an edge resource. To reach this objective, each module relevant to this use case must be implemented and properly configured. This includes the different modules of the local and global closed loops, as well as the communication system that connects them. In addition to the security system, which represents the core of this demonstration, a 5G system must be deployed, as it represents the vector and target of the attack. The different modules, as well as the implementation option chosen for this scenario, are represented in Figure 5-23. Note that the 5G core CP is not represented as it is not strongly relevant for this demo: all the attack takes place in the UP.

A sequence diagram representing the interactions of the different modules involved in this scenario is represented in Figure 5-24. As shown in this diagram, the different modules can be divided into three main categories: data path, local loop, and central loop. Kafka, which binds together all the modules of the local and central loops, is not represented in the diagram. In this section, the three categories will be detailed separately. However, it should be noted that the different interactions can happen in parallel. For example, the local loop starts to react to the attack as soon as the first attack packet is detected, and it does not have to wait for the attack to be completed.

The data path is the regular data path for a communication system without any specific autonomous security system. In this scenario, the functional behaviour of this category of modules goes as follows:

- The malicious traffic containing the log4shell attack payload is first generated by a regular UE registered in the network (1). This UE, as well as the RAN stack, are simulated by a customized version of the UERANSIM tool.
- The traffic will then reach 5G core UP, the UPF, implemented via VPP. The UPF forwards the traffic toward the IPS (2). At this point, the traffic is also mirrored into a dedicated Kafka topic (4). This monitoring traffic is consumed by the local loop.

- The traffic goes through the IPS and reaches its destination: the vulnerable app (3). At this point, the attack is successful. For demonstration purposes, the realization of the attack is materialized by the creation of a file in the vulnerable app.

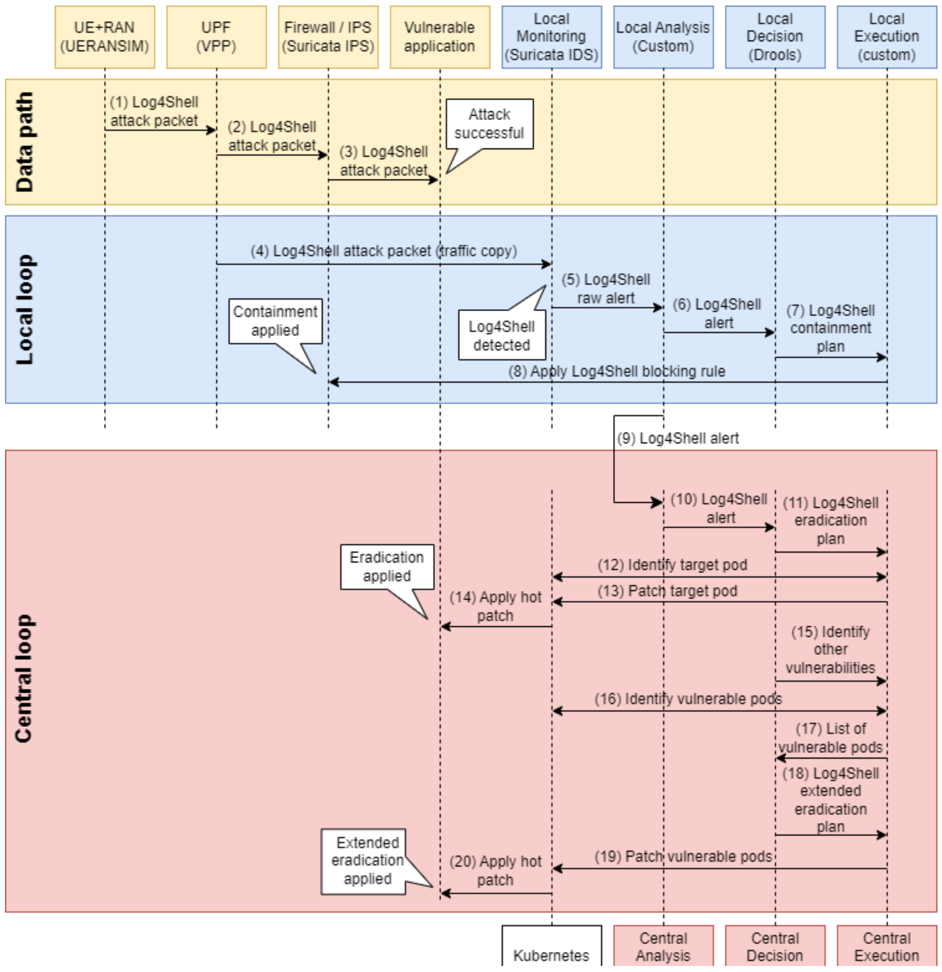

UE+RAN (UERANSIM)
UPF (VPP)
Firewall / IPS (Suricata IPS)
Vulnerable application
Local Monitoring (Suricata IDS)
Local Analysis (Custom)
Local Decision (Drools)
Local Execution (custom)
Data path
(1) Log4Shell attack packet
(2) Log4Shell attack packet
(3) Log4Shell attack packet
Attack successful
Local loop
(4) Log4Shell attack packet (traffic copy)
(5) Log4Shell raw alert
(6) Log4Shell alert
(7) Log4Shell containment plan
Containment applied
Log4Shell detected
(8) Apply Log4Shell blocking rule
(9) Log4Shell alert
Central loop
(10) Log4Shell alert
(11) Log4Shell eradication plan
Eradication applied
(12) Identify target pod
(13) Patch target pod
(14) Apply hot patch
(15) Identify other vulnerabilities
(16) Identify vulnerable pods
(17) List of vulnerable pods
(18) Log4Shell extended eradication plan
Extended eradication applied
(19) Patch vulnerable pods
(20) Apply hot patch
Kubernetes
Central Analysis
Central Decision
Central Execution

Figure 5-24. Scenario 5.3 sequence diagram

The data path is the regular data path for a communication system without any specific autonomous security system. In this scenario, the functional behaviour of this category of modules goes as follows:

- The malicious traffic containing the log4shell attack payload is first generated by a regular UE registered in the network (1). This UE, as well as the RAN stack, are simulated by a customized version of the UERANSIM tool.
- The traffic will then reach 5G core UP, the UPF, implemented via VPP. The UPF forwards the traffic toward the IPS (2). At this point, the traffic is also mirrored into a dedicated Kafka topic (4). This monitoring traffic is consumed by the local loop.
- The traffic goes through the IPS and reaches its destination: the vulnerable app (3). At this point, the attack is successful. For demonstration purposes, the realization of the attack is materialized by the creation of a file in the vulnerable app.

The local loop is a cybersecurity autonomous closed loop that monitors data path traffic, detects incoming attacks and applies containment actions upon attack detection. In this scenario, the functional behaviour of this category of modules goes as follows:

- Via a dedicated Kafka topic, the local loop receives a copy of the data plane traffic going through the UPF. This traffic is directly fed into the IDS (4). When the malicious packet reaches the IDS, a log4shell alert is raised and sent to the local Analysis module (5). The format of the alert depends on the IDS, here Suricata. While Suricata is used here as an IDS, other IDS could be used, either instead of Suricata, or in parallel.

- The local Analysis module format the alert and forwards it to the local Decision engine (6). At the same time, the alert is observed by the central loop (9). In more complex scenarios, the Analysis module would have a more important role, typically gathering several alerts to deduce the nature of the attack.
- The local Decision engine emits a containment plan for Log4Shell (7).
- The local Execution engine connects to the IPS via SSH. The private key of the IPS is pre-provisioned to the Execution module for this purpose. The local Execution engine applies the containment plan, which consists in adding a rule into the IPS to block any log4shell-realted signature. The rules are then live-reloaded. At this point, the containment measures are fully applied, and the UE cannot attack the vulnerable application anymore. Note that the rules were not present in Suricata IPS in the first place as the attack probability was considered to be low, and since the IPS is on the data path, it has to remain as lightweight as possible to avoid inducing delays to the UP traffic.

While the vulnerable app is now safe from an attack coming from UEs, the root vulnerability remains, and attacks from other sources (e.g., from a compromised application) remain possible. Consequently, eradication measures are required. The central loop is a cybersecurity autonomous closed loop that monitors local loops, detect incoming attacks and applies eradication actions upon attack detection. Generally speaking, the central loop could monitor any information generated by the local loop. Here, the central loop only monitors the alerts emitted by the Local Analysis module. In this scenario, the functional behaviour of this category of modules goes as follows:

- The central analysis module gathers the *log4shell* alert emitted by its local counterpart (9). As the attack is here straightforward, the module does not have to perform further analysis and can directly handle the alert to the central Decision engine (10).
- The central Decision engine emits an Eradication plan for log4Shell. This action is actually divided into two plans.
- Patch the vulnerable application (11)
- Find all other vulnerable applications within the domain (15) and patch them (18). This is the extended Eradication plan. The rationale behind this plan is that since the attack happened once, its likelihood in the domain increased and justifies preemptive measures.
- Upon reception of the patch request for the vulnerable application, the central Execution engine first determines which pod is running the vulnerable application (12) using the IP and PORT targeted by the attacker. Once the pod is known, the central Execution engine uses the K8s API (13) to apply the patch to the vulnerable application (14). At this point, the vulnerable application is no longer vulnerable to log4shell, and the eradication process is completed. To push further the resolution of the problem, the central Execution module can send a notification to the human administrator to suggest an update of the vulnerable application (step not represented in the sequence diagram). The update itself cannot be automated and requires the contribution of the developers of the application.
- Upon reception of the vulnerability identification request (15), the Central Execution module uses its K8s access to identify other vulnerable application pods (16). This identification is performed by the Trivy vulnerability scanner which is able to detect the *log4shell* vulnerability in container images. The list of vulnerable images is sent back to the central Decision engine (17).
- Upon reception of the subsequent patch request for that application (18), which constitutes the extended Eradication plan, the local Execution engine applies the patch to the vulnerable pods. This back-and forth-between Decision and Execution engines demonstrate the ability of the system to perform additional investigations if required. This specific example consists of not only applying the patch to the directly affected application but extending the procedure to other applications. Such back and forth could also take place between the response block and the decision & analysis one, for example, if the existence of a specific attack or step of attack may involve the existence of other attacks that should be looked for.

- All the messages between the different entities are exchanged through Kafka topics (when it is not specified otherwise, for SSH connections, for example). As said previously, this allows the later addition of new monitoring/analysis/ decision/actuation engines to cover a wider variety of attacks and provide a complete overview of the security situation system to the administrator. Likewise, Kafka enables other components, for instance, other security functions, to consume the information generated by local and central closed loops to support their decisions or actions to be taken. For the sake of simplicity, all topics are handled here by a single Kafka system; however, the different topics are well isolated and could be handled by different Kafka instances to match the general architecture presented in Section 5.3.3.1. The information produced during the local and central loops may be consumed by other security functions instantiated through Service, Network or Infrastructure Layers of the M&O architecture design [HEX22-D62]. In this specific scenario, the LoTAF, under the Service Layer, leverage outputs coming from the analysis, decision and execution modules of local and central loops. The principal objective of LoTAF is to demonstrate how real-time cyber security threats, like the log4shell, may have an impact on the Service Provider (SP) LoT and the agreed security and privacy requirements.

Since all the messages between the different entities involved in local and central loops are exchanged through Kafka topic, LoTAF needs to deploy a Kafka Consumer to subscribe to the topics it is interested in, specifically, *local_analysis*, *central_analysis*, *local_decision*, *central_decision.* Due to the fact that the local loop carries out quicker and simpler countermeasures than the central loop, the LoTAF needs to be informed of all types of countermeasures that are applied when a threat appears. First and foremost, the LoTAF recaps the information of the Local Analysis module to find out whether the current threat is one of the possible threats detected by LoTAF at Stage 1. As previously mentioned, the likelihood of the log4shell attack has not been considered severe enough, so it was not under the possible threats of the instantiated network service. In consequence, the next step of LoTAF is to figure out how a new threat may compromise the security and privacy requirements agreed upon between an end-user and an MNO. In this specific scenario, LoTAF assumes the Detection, Analysis and Remediation modules were instantiated before the event. Therefore, there are no signals that security and privacy requirements were compromised, but they might be compromised if containment and eradication plans are not fully effective. In this vein, the containment plan linked to the Local Response module makes use of blocking the traffic that matches the *log4shell* signature by installing the proper rules in Suricata IPS. This containment action aims at dwindling and stopping the log4shell threat by dwindling the likelihood of suffering such an attack. Yet, the vulnerability is still in the application or applications (meaning of yellow colour in ovals); therefore, the LoTA should assess the feasible risk.

Figure 5-25 displays, on the right side, the security and privacy requirements that the MNO needs to ensure to end-user (ovals) as well as a set of countermeasures related to the local closed loop. Note that the pink boxes represent both the loop and the countermeasure being used at this moment. In order to reassess the LoT, LoTAF makes use of a reward and punishment method to adjust it based on the selected containment and eradication plans. As part of this method, LoTAF also leverages a rule-based decision engine to determine countermeasure effects in the initial LoT as well as reassessment-based learning techniques. As a result, the reward and punishment method determines the affinity (membership degree) between a pre-defined set of thresholds, the countermeasure effects, security and privacy requirements previously agreed and optimization functions. Figure 5-25 displays, on the left-hand side, the affinity of the new LoT with the thresholds using a trapezoidal fuzzy model. In particular, the new LoT has a membership degree of 0.6 with the Trustworthy level and 0.38 with the Moderately Trustworthy level.

Owing to the fact that the central closed loop has not still applied the eradication plans, which guarantee logh4shell attack is not currently a threat to the running applications under the network service, LoTAF makes a conservative decision when evaluating LoT. Thus, it considers both the likelihood that current filtering rules might not be effective for future deltas of the log4shell attack and the impact of the attack on the end-user requirements. Therefore, the LoT is updated to the

Moderately Trustworthy level so as to watch out if integrity, confidentiality or availability requirements might be compromised until the eradication plan is finally applied.

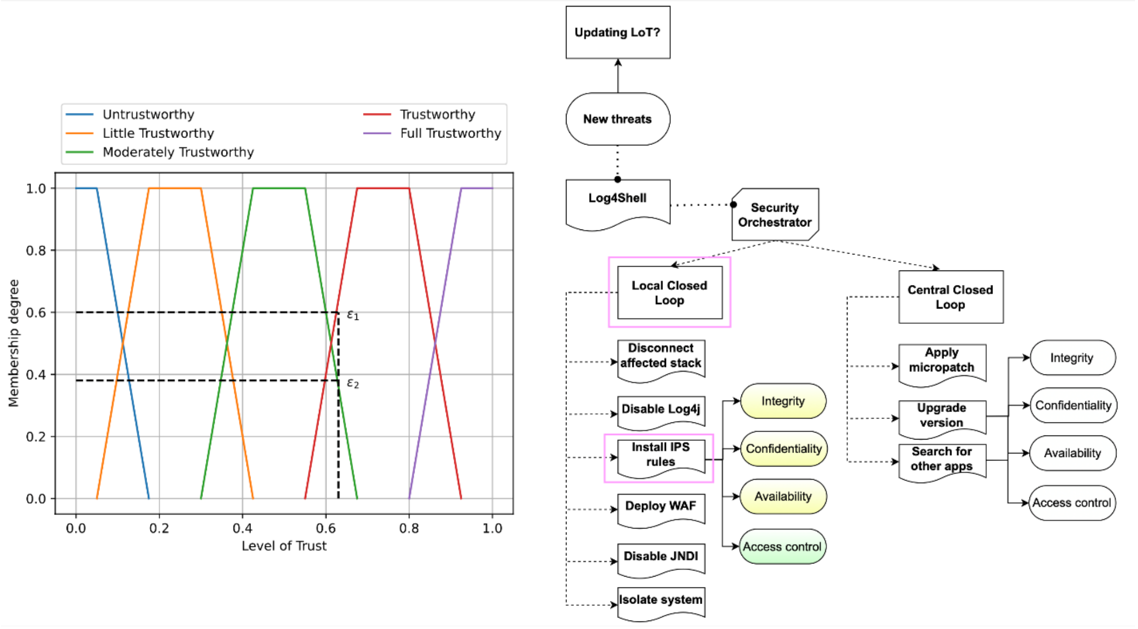


Figure 5-25. Level of Trust update based on containment plan for Scenario 5.3.

Since there is no defined time until the eradication plan is finally applied, because it requires more complex and consuming-time tasks than the containment plan and even the administrator interaction to perform a specific patch or a full upgrade of the applications, the LoT is updated after eradication actions are carried out. To this end, the LoTAF will apply the same reward and punishment method, but this time the countermeasures, the likelihood and the impact of log4shell are totally different. In this sense, the rule-based decision engine and the reassessment-based learning techniques determine that the risk of suffering such a threat has been properly tackled (meaning of green colour in ovals), and in consequence, the LoT is slightly increased until reaching the initial Trustworthy Level (see Figure 5-26).

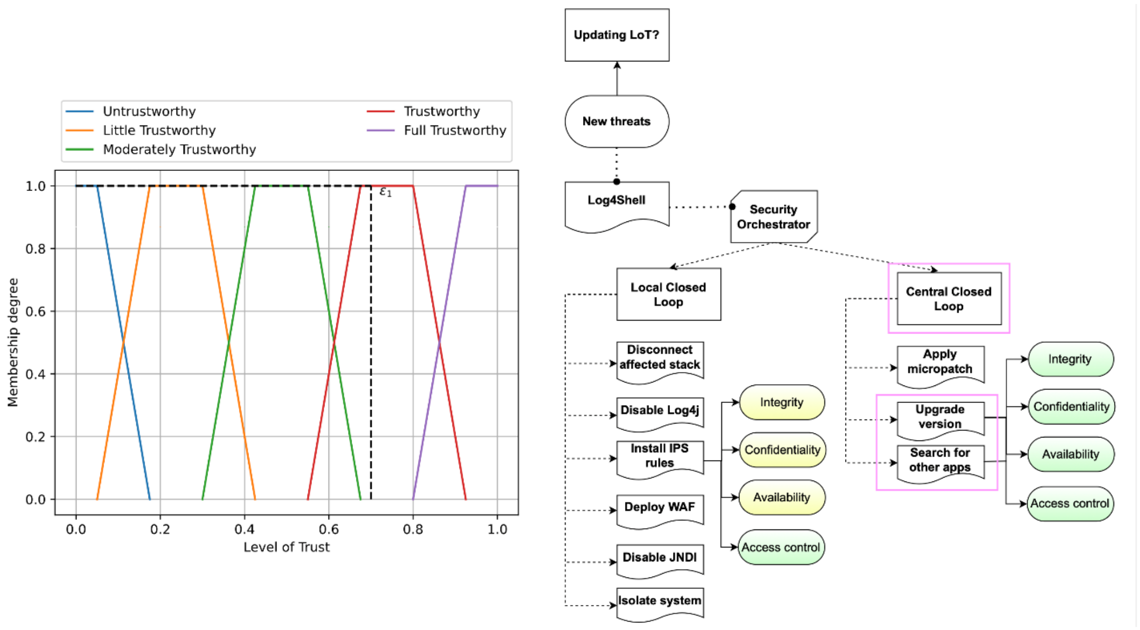


Figure 5-26. Level of Trust update based on eradication plan for Scenario 5.3.

5.3.3.4 Deployment

As represented in Figure 5-23, the system relies on two management and orchestration tools: OpenStack and K8s. K8s itself is installed on VMs managed by OpenStack. Table 5-4 details the requirements of the different components.

Table 5-4. Scenario 5.3 VM-based component list.

Name	Function	Type	vCPUs	RAM (GB)
UERANSIM	UE + RAN simulator	VM	2	4
UPF	5G User Plane	VM	4	8
Suricata IPS	Firewall	VM	4	8
Master	K8s master node	VM	4	8
Worker-1	K8s worker node	VM	8	20
Worker-2	K8s worker node	VM	8	20

The deployment is made manually: VMs are first created following size requirements. Then, components based on VMs (K8s included) are installed, and finally, containerized components are deployed using Helm charts. Table 5-5 details how the different components are installed – the containerized components do not require additional resources.

Table 5-5. Scenario 5.3 container-based component list.

Name	Function
Local Suricata IDS	Local Monitoring
Local analytic module	Local analysis
Local Drools	Local decision engine
Local execution	Local execution
Central analytic module	Central analysis
Central Drools	Central decision engine
Central execution	Central execution
Kafka Broker 1	Communication between components
Kafka Broker 2	Communication between components
Zookeeper	Support for Kafka
Vulnerable application	Target application to demonstrate the attack
LoTAF	Level of Trust Assessment Function

5.3.4 Scenario 5.4: MLOps techniques to deploy AI/ML service components

5.3.4.1 Scenario description

MLOps is a concept devised to refer to the full lifecycle management of ML (and its variants, Deep Learning, Reinforcement Learning, etc.) in production[7]. In short, MLOps can be understood as a particularization of the already well-known DevOps paradigm [EAD14], aimed at addressing the challenge of developing and deploying in production AI/ML-based software artefacts with an approach similar to that used in DevOps. The main challenge that the MLOps approach seeks to

[7] The MLOps term was originally coined by Dr. Nisha Talagala back in 2018 (see https://www.linkedin.com/in/nisha-talagala-6a6b20, https://www.forbes.com/sites/nishatalagala/?sh=4b2ac4b63de9 or https://www.slideshare.net/NishaTalagala/ml-ops-pastpresentfuture.

address is the integration of the processes of collecting and formatting the training data typically needed by the AI/ML models, as well as the training process itself, as these processes are not considered in the regular DevOps workflows[8].

The MLOps concept comes from the IT industry. However, the target in the context of this Hexa-X project is to explore how this concept could also be applied in the telco-grade industry, specifically, regarding the future 6G systems, wherein AI/ML techniques are expected to play a relevant role. In this regard, it is especially relevant the integration of the different stakeholders typically operating in the telco sector. Scenario 5.4 intends to take an initial step in this direction, trying to showcase how the MLOps practices could be applied in the scope of telecommunications. The scenario considered in this demo targets a specific use case where a single SW Vendor develops, trains, and deploys an AI/ML-based model on the MNO infrastructure[9]. Specifically, a supervised learning model has been chosen to showcase the whole MLOps cycle, trying to address the problem of sharing the data needed to train the model between two different administrative entities: the SW Vendor and the MNO. In particular, the ML model used in the demo aims at providing AI capabilities in network management and orchestration, specifically to avoid network slice and service performance degradations caused by limited UPF resources at the edge [HEX23-D43]. The model serves for the optimal auto-scaling of UPFs placed at the network edge in support of low-latency communication services. The scenario also covers a simple model drift management use case, which automatically redeploys the model when a drift situation is detected. Figure 5-27 shows the overall approach for this scenario, representing both the SW Vendor Domain (left) and the MNO Domain (right), with the latest composed of two different environments, staging and production.

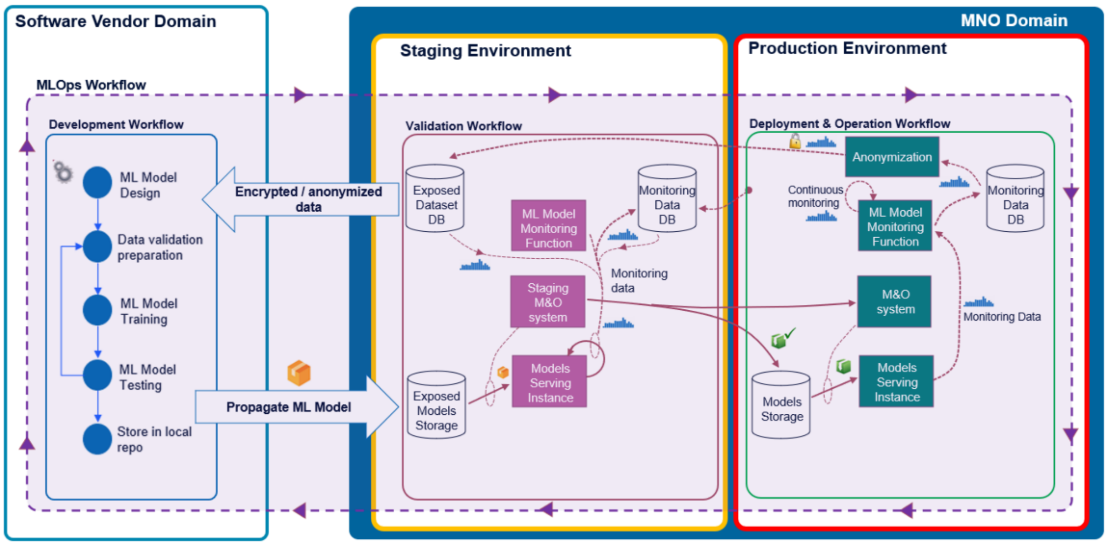


Figure 5-27. Scenario 5.4 block diagram.

These domains and environments have been instantiated on different VMs to simulate the separation between these domains and environments that would actually occur in a real-life scenario. As can be seen, the overall MLOps workflow (purple dashed line) consists of different sub-workflows in the SW Vendor Domain (the Development Workflow) and the MNO Domain (the Validation Workflow in the Staging Environment and the Deployment and Operation

[8] Please, be aware that MLOps (the topic addressed here) is not the same as AIOps (another commonly related topic). In short, AIOps can be considered as the application of AI/ML techniques to DevOps, while MLOps would be the application of the DevOps methodologies to develop and deploy AI/ML-based artifacts [Ler17].

[9] To consider just a single SW Vendor is indeed a simplification made in the context of the demo. It is well known that in the telco-grade environment this is not always the case, as network services to be deployed on the MNO infrastructure are often developed by different vendors. However, though it is considered that such multi-vendor scenario could be interesting for future research (it would require more complex workflows taking into account the coordination of the different suppliers), it has been considered better to start with this simplified approach with just a single vendor, as an initial step in the context of this Hexa-X project regarding the MLOps approach.

Workflow in the Production Environment). The Development Workflow is intended to simulate the development stage that occurs at the SW Vendor domain, and that, in this case, also includes the AI/ML-model design and training phases. The Validation Workflow occurs already at the MNO Domain, in its Staging Environment, and basically comprises the verification phase of the model delivered by the vendor, which is always necessary prior to its acceptance and deployment in the production environment. Then, the Deployment and Operation Workflow covers the necessary tasks to put the model in production (once validated) and its continuous monitoring. These workflows, together with the means that have been used for their implementation, is described in more detail in the corresponding subsections below.

5.3.4.2 Functional Behaviour

This subsection addresses the description of the overall MLOps workflow itself, which, as said, consists of different *sub-workflows*: one of them executed at the SW Vendor Side (the Development Workflow), and the other two on the staging and the production environments of the MNO (Validation Workflow and Deployment & Operation workflows respectively). In the following, each of these sub-workflows is described.

SW Vendor Side Workflow

As mentioned, the Development Workflow targets the AI/ML model development itself (being the equivalent of the "Dev" part in the regular DevOps approach). Of course, it is assumed that, in a real-life scenario, prior to the execution of this workflow, there would be a Service Level Agreement (SLA) set up between the MNO and the SW Vendor. As for regular non-AI/ML-based services, that SLA would come after a negotiation process between the two parties, after which the requirements of the service (based on an AI/ML model in this case) should be well defined. For the demo, we will omit this SLA development stage, which is considered to have already occurred before the model development work itself begins. As can be seen in Figure 5-27, this Development Workflow consists of seven stages, namely:

1. **Get anonymized/encrypted training data from the MNO scope**. This step has been represented in the figure by the right-to-left arrow entering into the SW Vendor Domain. In a real-life scenario, this could be understood as a recurrent process having multiple iterations. The first iteration would be the initial communication process between the Vendor and MNO operational teams, where that second party would communicate the necessary initial data to start designing, developing and performing the first training tests on the AI/ML model. Probably, these 1st iterations would be a mix of offline and online communications between the Vendor and MNO. However, in later iterations, once the data model is well established, that communication process could happen in a more automatic way or even could be fully automated. For the demo implementation, the process has been simplified by defining a fixed data model and using a defined data set that the MNO transfers to the Vendor. As can be seen, the communication is performed by exposing the MNO Datasets DB (see Figure 5-27), in line with the API Management Exposure concept introduced in the M&O architectural design from the previous Deliverable D6.2 [HEX22-D62]. As mentioned above, for real-life scenarios, the fact that the training data exposed to the Vendor must be anonymized or encrypted would be of utmost importance since the MNO would be obliged to keep the user data confidentiality (e.g., by the GDPR [GDPR], or by specific obligations with other parties). However, the training of the model shall be possible even when the data is encrypted and/or anonymized so that, once trained, it can make inferences in the MNO environment by using the non-anonymized (or encrypted) data once in production. The demo targets specifically this approach, which is considered relevant in the telco-grade environment for those cases where the training data from the MNO must be shared with external parties (e.g. when using the supervised and/or unsupervised learning paradigms). Specifically for the demo, the Micro-aggregation (MA) method has been used before storing the training data on the Exposed Dataset DB. This technique makes it possible to train the models using anonymised data [STO+20].

2. **ML Model Design**. In this step, the AI/ML-model development team decides on the specific AI/ML approach to be used to better solve the design requirements. In a real-life case, at this stage, it is decided, for example, which neural network topology to use, the specific learning algorithm to apply, the size and composition of the data sets for training and testing, etc. Of course, typically, this happens only at the beginning of development. In future iterations of the MLOps workflow, this can be omitted unless a complete re-design is necessary. For the demo, this stage has been simplified, considering a network topology already pre-designed. Specifically, a Long Short-Term Memory (LSTM) flavour model was used, consisting of a graph convolution layer followed by LSTM and dense layers. Evaluation of the model performance was done through the symmetric Mean Absolute Percentage Error (sMAPE) on test data.
3. **Data Validation and Preparation**. This stage comprises the filtering and normalization of the data that is typically necessary prior to the AI/ML model training. For the demo, for the data validation, an analysis of the input data is performed, checking out anomalies in the provided data.
4. **ML Model Training**. Here is where the model is trained to perform the requested function. In this case, an LSTM model [SRO+19] is trained using time series data to predict the future state of the UPF load.
5. **ML Model Testing**. This is the testing stage. As can be seen in Figure 5-27, depending on the testing results, the previous data validation/preparation and the training itself could be repeated over and over until the results are according to the requirements in the SLA. In the demo, this stage has been implemented by means of setting constraints on the evaluation phase of the pipeline.
6. **Store in the local repo**. This step just represents the successful finalization of the previous testing process and the storing of the generated AI/ML model in the SW Vendor's local repositories. The model is stored there until its propagation to the MNO scope.
7. **Propagate the model towards the MNO scope**. This is the last step in the Development Workflow, represented by the left-to-right arrow getting out from the SW Vendor Domain in Figure 5-27. However, although a “last step” in the Development Workflow, this is just an intermediate step in the whole MLOps workflow that, in fact, can be executed as part of multiple MLOps cycles. As with DevOps, the goal here is also to perform Continuous Delivery, propagating new versions of the model as soon as they are made available by the vendor in a highly automated way. Although in a real-life scenario, the initial deliveries of the model could probably be semi-automated or even manual, the objective here is to automate the process as much as possible in such a way that deliveries can be performed in a continuous way. As it can be seen in Figure 5-27, the delivery is performed using an *Exposed Models Storage*, also following (as with the getting for the training data exposure in step 1) the API Management Exposure concept introduced in [HEX22-D62], and specifically, by exposing to the SW Vendor the access to the Exposed Models Storage REST APIs (to store and later update the models, the training pipeline artefacts, or any other relevant metadata).

MNO Side Workflows

Figure 5-28 shows the main steps of the workflows executed at the MNO domain, happening in the MNO Staging Environment (the Validation Workflow) and in the Production Environment (the Deployment & Operation Workflow). As can be seen, both environments (staging and production) contain basically the same set of components:

- The **Models Serving Instance**, which basically provides the “execution environment” for the AI/ML model developed and used in the demo.
- The **ML Models Monitoring Function**, monitors the performance of the ML models, evaluating their runtime accuracy. It includes methods for identifying potential drifts in the model inferences as well as in the inference data.
- The **Monitoring Data DB** is used to store and maintain different types of data relevant to the MLOps cycles: the training data, the inference runtime data, and the model

monitoring and evaluation data. In practice, it stores all the required data to implement (on the Vendor side), validate (at the MNO staging side), serve (on the MNO production side) and monitor/evaluate (on the MNO production side) the models. Specifically, the Monitoring Data DB is the main source of data for the Exposed Dataset DB.

- The **M&O System** is used to orchestrate the MLOps workflow in each MNO environment.

Besides these main components, there are also the two exposed databases in the staging environment described in the previous section (those used to send training data to the SW Vendor and to receive the models from it) and also, the Models Storage DB in the Production Environment, which basically mimics the functionality of the Exposed Models Storage in the staging environment, but within the MNO production domain itself. Another difference is in the Anonymisation Component, which, as can be seen, is deployed only in the production environment. The duplication of components in staging and production environments is obviously intentional, trying to reproduce what actually happens in real life to provide a testing environment as similar as possible to the production environment. From a conceptual perspective, this MLOps approach with separation of staging and production environment is applicable to real-life scenarios where the MNO hosts multiple production environments (e.g., to support different types of services or customers). The Validation Workflow (the 1st workflow executed after the AI/ML model is received from the vendor) implements the validation process that typically takes place in real life for the network services delivered to the MNO by external parties. AI/ML-based services are not different to this in the regard that they also need to be validated before their deployment in the production environment. The steps of this Validation Workflow are encircled with the numbers '1', '2' and '3' in Figure 5-28.

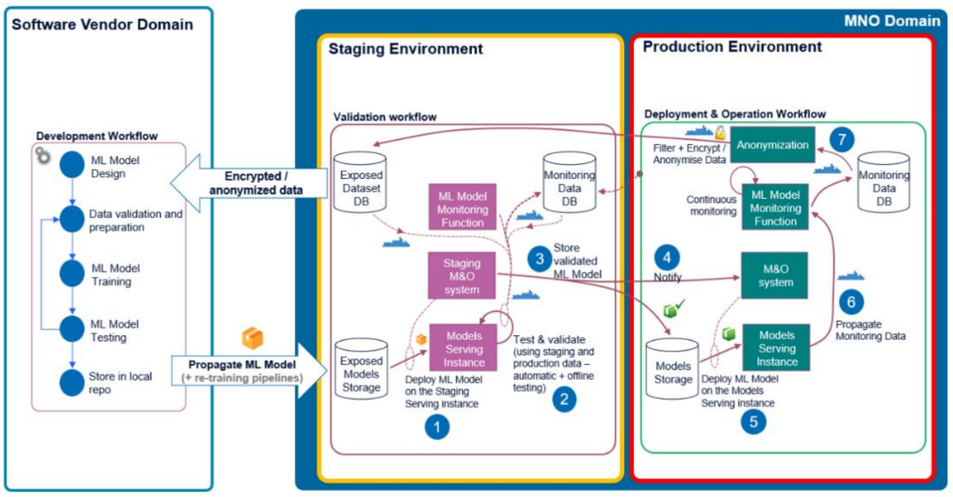


Figure 5-28. Main steps of the MLOps workflow in the MNO Domain.

As can be seen, the first step consists of the deployment of the model from the Exposed Models Storage on the Models Serving Instance. This is triggered by the Staging M&O System (dashed line). The objective here is to have the AI/ML-model deployed in an execution environment similar to the production environment to perform the necessary validation and testing on it (step 2) to accept the model or inform the SW Vendor in case some updates are required (this last interaction is not represented in the diagram, but it would basically consist on re-executing the Development Workflow at the SW Vendor Side to re-engineer the model to avoid drifts or faults found during the testing). The performing of Step 2 (testing and validation) is done differently depending on the maturity of the delivered model. In the initial stages, human-conducted testing (i.e., manual testing) could be necessary. However, in later iterations, this process could be highly automated using testing automation tools with the appropriate collection of test batteries. In the demo, this testing process has been implemented through the ML Monitoring Function, which basically evaluates the model accuracy for a given number of inferences (i.e., UPF load predictions) by measuring how much the prediction differs from the actual future data in the Monitoring Data DB and detecting potential drifts based on pre-defined thresholds. The results of the validation are stored in the Monitoring Data DB for manual inspection; however, the choice of validation merit is determined automatically by the ML Monitoring Function using a threshold, and the information is passed to the Staging M&O system for action. In any case (manual or automatic), this testing stage in the workflow is intended to be based on testing data from two different main sources:

- The data in the Exposed Dataset DB, i.e., the same data that is provided to the SW Vendor for it to generate the model.
- The Monitoring Data DB in the Staging Environment (top-right DB in the figure). This DB is intended to contain monitoring data, not only from the staging environment itself but from the production environment also. This data coming from the production environment is used to test the response of the AI/ML model under realistic service circumstances and not only using staging data or the datasets shared with the SW Vendor to train and validate the model. That production data can be stored in this Staging Monitoring Data DB through automatic data collection processes (from the Monitoring Data DB on the production side) or be manually stored by the MNO operations team members.

Step 3 happens once the testing process in the staging environment has been successfully completed. It basically consists in storing the validated AI/ML model in the Models Storage database, already in the production environment. Together with this, the M&O System at the production stage also receives a notification (step 4), meaning that a new AI/ML model (or a new version of an existing model) is ready to be deployed in the production environment. This notification shall have associated the information and metadata necessary to orchestrate the just released AI/ML-model, such as the model name and version, its main features and capabilities, its data requirements, its validity execution times, relation to other models, or whatever other metadata is necessary to properly execute it. In the context of the demo, the model tested and validated in the staging environment is stored in the production Model Storage DB by the staging M&O system through the exposed database REST APIs. On the other hand, the staging M&O system notifies the production M&O system by using a publish/subscribe mechanism based on a message bus.

Once the AI/ML model is deployed in the production Models Storage DB, the Deployment & Operation Workflow at the production environment can start. The 1st step in this environment (Step 5 in Figure 5-28) is the deployment of the to make the model available to the production Models Serving instance, which in practice will mean putting the model into production. As in the staging environment, this is triggered here by the production M&O system (dashed line). In real life, this step could be done under human supervision but also fully automated, targeting the Continuous Deployment paradigm. This second case, however, would be valid only for certain well-defined circumstances, which would be well defined in the orchestrator. For the demo, the production M&O system takes care to properly configure the Model Serving instance in order to have access to the proper model version. This is done in accordance with the metadata available in the Models Storage DB and which describes the capabilities and characteristics of the model. As soon as the AI/ML model is deployed, a continuous monitoring process is started driven by the ML Monitoring Function. This function has been developed from scratch and is executed as a containerized function. The Monitoring Function continuously takes relevant metrics from the deployed model (step 6 in the figure) and stores them in the Monitoring Data DB. It also compares the previously predicted output of a model with the real-time data for UPF load to check for drift, as described below under Drift Management. Those metrics are also anonymized and sent to the Exposed Data DB in the staging environment (step 7) in order to make them available for future testing on that environment and also to share them with the SW Vendor in case a new re-design and/or re-training iteration were necessary. For the anonymization process, the Micro-aggregation (MA) method has been used. In MA, first, the original values of a given dataset are partitioned into micro-clusters, and then they are replaced by each micro-cluster's centroid value. The basic idea is to generate homogeneous clusters over the original data in a way distance between clusters is maximized in order to minimize the information loss. One of the main reasons of using this method is that allows making inference on the trained model without applying the MA method on the inference data [STO+20].

Drift Management

The monitoring process mentioned in the preceding paragraph is aimed at detecting any malfunction of the AI/ML model, either to alert the MNO operations team (e.g., by an alarm signal on a console) and/or to trigger the appropriate automatic response to solve or minimise the effects of the failure. In AI/ML models, a common source of failures is what is typically known as “drift”, which refers to the loss of accuracy of AI/ML models that can occur when the data to which the model is exposed in the production environment differs from the dataset used for training. This, of course, can negatively affect the business, so it should be avoided. Drift can happen for different reasons, e.g., because of changes in the way users behave or use the network services because the MNO tries to use the model in a new context or due to a lack of precision in selecting the training data sets. MLOps can help to overcome this situation by implementing the appropriate drift detection and remediation strategies that could be executed in a highly automated way. In the demo, a specific use case addressing an example of drift management has been implemented as part of the Deployment & Operation workflow previously mentioned. Figure 5-29 shows this specific approach for the demo (steps ‘a’ and ‘b’ in the figure) that simply assumes that there is an alternate version of the AI/ML which can be used to remediate the drift situation. The use case is simple: the ML Model Monitoring Function detects the drift situation (a) and triggers the M&O system, which in turn causes the deployment of the alternate AI/ML model. Although simplified, this approach can be valid for real use cases in the telco-grade environment where, for instance, there may be different service usage patterns depending on the time (e.g., time of day or day of the week). Based on this, the Models Storage DB could have different versions of the AI/ML model, each suited to a specific situation. In the demo, a drift is detected through continuous comparison of the previously predicted output of a model with the real-time data for UPF load. This is handled through the ML Monitoring function. When the Monitoring detects a degradation in the prediction accuracy, it passes the information to the M&O system for action. The M&O system then triggers the evaluation of the available model versions, similar to the model validation process described in Step 2 above. When a model version is found which meets the requirements of the SLA, the M&O system deploys the model to production, as described in Step 3. It is not included in the demo, but if no suitable model version is found, the M&O system will trigger a re-training of the model. If the SLA cannot be met by the newly trained model, a message is sent to the SW vendor to request a new model development.

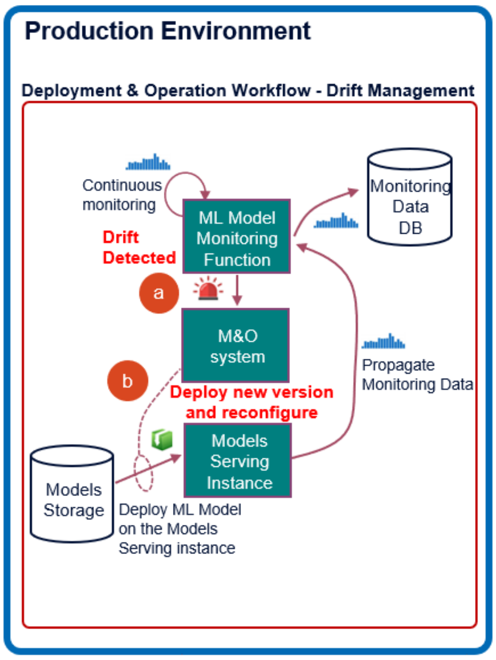

Production Environment
Deployment & Operation Workflow - Drift Management
Continuous monitoring
ML Model Monitoring Function
Monitoring Data DB
Drift Detected
a
M&O system
b
Deploy new version and reconfigure
Propagate Monitoring Data
Models Serving Instance
Models Storage
Deploy ML Model on the Models Serving instance

Figure 5-29. MLOps Workflow - Drift Management.

Of course, other more complicated drift management scenarios could be considered. E.g., there could be considered a re-training of the model in the staging environment (aligned with the Continuous Training concept [TBF+22]) or even to request for the development of a completely new version of the model to the SW Vendor (e.g., through the MNO BSS systems). However, in terms of demonstration, it has been considered that the simplistic approach in Figure 5-29 is far enough for an initial step to introduce this concept.

5.3.4.3 Implementation Details

Figure 5-30 shows the specific software components that have been used to implement the demo; below is a description of these components.

SW Vendor Domain

Four main components have been used here:

- K8s. In the context of the demo, this is used as it allows to scale and manage containerized software components and execute them as cloud-native applications. An additional reason is that the orchestration platform described in the next paragraph can only run on top of K8s.
- Kubeflow. This is an open-source system for making deployments of ML workflows on K8s [KUBb]. In short, it offers an ML toolkit where the SW Vendor workflow pipeline can be developed, executed, and tested. In the context of the demo, it has been used to orchestrate the Development Workflow itself, as it offers a machine learning toolkit not only for orchestrating pipelines but also to develop them with notebooks servers and katib components, or even make experiment tracking in contrast with other orchestrating tools like Apache Airflow [AIR]. Additionally, it allows multi-user isolation by creating the resources in different namespaces for each user. The orchestration function itself has been implemented by the orchestration component provided by Kubeflow, called “Kubeflow Pipelines” [KUP].
- The MinIO database [MIN]. This is the vendor's local database just mentioned. It is a K8s-native objects storage.
- TensorFlow Extended (TFX)[TFX] is an extension of the well-known TensorFlow framework [TFL], which is typically used to create and manage ML production pipelines. As can be seen in Figure 5-30, TensorFlow Extended is used in the demo to implement three of the steps in the Development Workflow, namely:
  - The Data Validation and Preparation step is performed by means of the default TFX pipeline components oriented to make this preparation and validation. The first one is the ExampleGen component which ingests the data into the TFX pipeline; it is the first component present in the pipeline. The second is StaticsGen which generates features statics over the ingested data. The third is SchemaGen, which is in charge of generating a schema with information about the types, categories and ranges from the training data. Finally, the ExampleValidator compares the data statics generated by the StaticsGen component against the schema produced by the SchemaGen component to detect anomalies in the input data. All these aforenamed components make use of the TensorFlow Data Validation [TFDV] library.
  - The ML Model Training step, which is performed by the Trainer component, takes three mandatory inputs. The examples generated by the ExampleGen, a module file that defines the trainer logic that is basically a python script where the model is defined, and finally, a definition of the trainer args (e.g. the number of training steps).
  - The ML Model Testing step, which is performed by the Evaluator component, which takes as input the trained model produced by the Trainer component and the data from the ExampleGen component. This component allows setting statistical metrics such as Mean Squared Error, Accuracy etc. It also helps ensure that the model is "good enough" to be pushed to a production environment. To that end, the component uses the TensorFlow Model Analysis [TFMA] library to perform the analysis.

As seen in Figure 5-30, the initial and final steps in the Development Workflow (i.e., the ML Model Design and the Storage in the Local Repo) are outside the TensorFlow Extended framework. The initial one is obviously because the design is typically executed in an offline manner by a design team. In the demo, a pre-designed model is used. The final one (the storage in the local repository) is just a simple commit operation that, in the demo, is executed by simply storing the generated model in the local database.

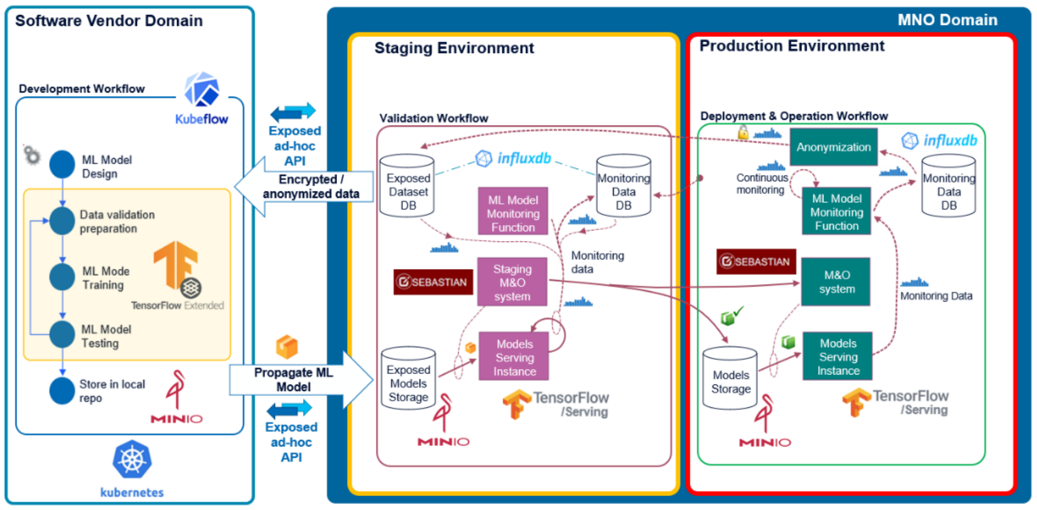


Figure 5-30. MLOps Scenario functional blocks and software components.

MNO Domain

The following software components have been deployed here:

- MinIO. The same objects' storage as in the SW Vendor Domain. In this case, two different instances have been deployed: one in the Staging Environment and another one in the Production Environment. The first one (implementing the Exposed Models Storage) is used to store the ML models provided from the vendor side, while the second one (the Models Storage) is used to store those models that have been already validated in the Staging Environment. As shown in Figure 5-30, the Exposed Models Storage relies on an Exposed ad-hoc API to allow the vendor to store the AI/ML models on it. This ad-hoc API has been implemented using the python client API provided by MinIO, which supports uploading files into a specific bucket with previous authentication making use of access and secret keys generated by the MNO.
- InfluxDB [INF]. This is another database, specifically a time-series database, that is used for different purposes: This is another database, specifically, a time-series database that is used for different purposes:
    - To implement the Exposed Dataset DB in the staging environment, i.e., the DB the MNO uses to share the necessary training data with the external SW Vendor. To make the vendor able to access its data, an Exposed ad-hoc API has also been implemented here. In this case, this ad-hoc API uses the HTTP API provided by InfluxDB that allows writing on buckets and query data; as with the MinIO instance, authentication is required.
    - To also implement the Monitoring Data DB in both environments: staging and production.

  In all these cases, the usage of a time-series database is justified because the type of data collected are asynchronous timestamped metrics, and the downsample and aggregation functionalities of a time-series database can be exploited to reduce the computation needed during data preparation stages.

  To implement the Exposed Dataset DB in the staging environment, i.e., the DB the MNO uses to share the necessary training data with the external SW Vendor. To make the vendor able to access its data, an Exposed ad-hoc API has also been implemented here.

In this case, this ad-hoc API uses the HTTP API provided by InfluxDB that allows writing on buckets and query data; as with the MinIO instance, authentication is required. To also implement the Monitoring Data DB in both environments: staging and production. In all these cases, the usage of a time-series database is justified because the type of data collected are asynchronous timestamped metrics, and the downsample and aggregation functionalities of a time-series database can be exploited to reduce the computation needed during data preparation stages.

- **Anonymization Component**. This component performs anonymization over the dataset which is used by the SW Vendor Domain. This application is hosted in a K8s pod between the Monitoring Dataset DB and the Exposed Dataset DB in order to deliver the anonymized data into the Exposed Dataset DB. The application uses the Micro-aggregation (MA) method [STO+20] that performs data anonymization allowing to use of the data to train ML models.
- **TensorFlow Serving [TFS].** This is a flexible serving system for ML models. For the demo, this is a sort of "execution environment" where the AI/ML models are deployed and are made available for inferencing requests through an API (in the context of the demo, a REST API is used). As can be seen in Figure 5-30, two identical instances are deployed for the staging and the production environments, respectively. This component is executed as a containerized function.
- **AI Agent**. This module, along with TensorFlow Serving, is part of the Model Serving instance. It retrieves the run-time data from the Monitoring Data DB and sends a request for inference to TensorFlow serving. It also contains the logic for predicting a need for altering the resources allocated to the UPF. This information is passed to the M&O for action. This component is executed as a containerized function.
- **The M&O System**, which is based on the open-source Vertical Slicer software [5GR21-D24], was developed by Nnextworks for the management of vertical services and end-to-end network slices across RAN, core and transport domains. A new dedicated functionality for managing and orchestrating the AI functions used in this demo has been implemented. Two instances are deployed (in the production and the staging environments), which are used to orchestrate the different steps of the Validation Workflow (in the Staging Environment) and the Deployment and Operation Workflow (in production).
- **ML Model Monitoring Function**: This custom component performs the continuous monitoring of the deployed model's accuracy. As described above, it uses a comparison of the previously predicted output of a model from the AI Agent with the real-time data for UPF load. This component is executed as a containerized function. As can be seen in Figure 5-30, there are two instances for the staging and the production environments, respectively.

Regarding the deployment of the demo, it has been done according to what has been represented in Figure 5-31.

As seen, the deployment has been done on two physical nodes (NUC1 and POP3-Amanita in Figure 5-31), representing the two separated domains in the demo: the SW Vendor Domain and the MNO Domain. On the SW Vendor Domain, the "hexax-atos" K8s namespace has been created to host the components to implement the Development Workflow in the form of a local Kubeflow pipeline. On the other hand, in the MNO Domain, two VMs have been instantiated: one VM hosting the Staging Environment components to implement the Validation Workflow (NXW-AI-STAGING in the figure), and another VM implementing the Development and Operation Workflow (NXW-AI-VM)[10]

[10] There is in fact a third VM in this MNO domain (NXW-UPF-VM) hosting the UPF component itself, which provides runtime data for the AI/ML model deployed by means of the MLOps workflow explained through this section. However, this component has been left out of the diagrams, as it plays no role in terms of the MLOps workflows itself.

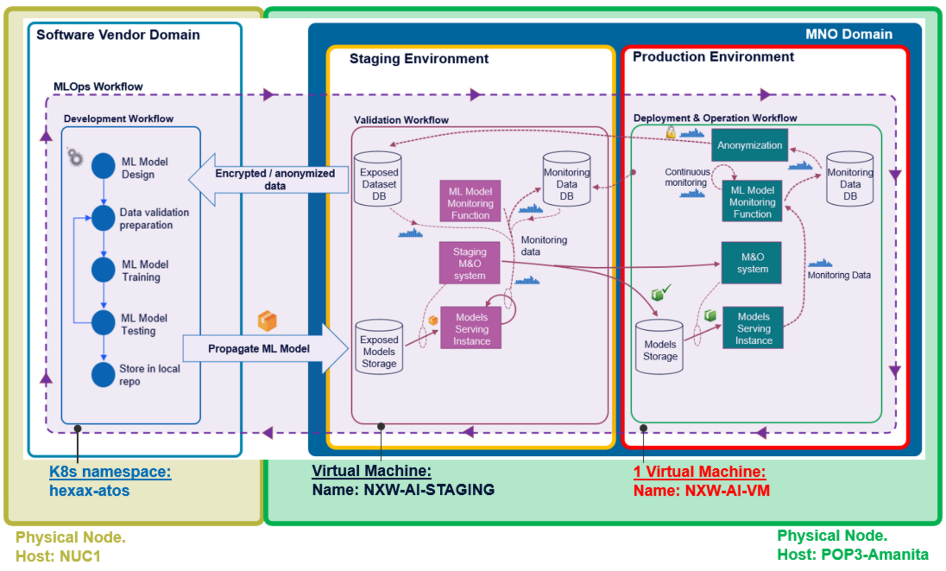


Figure 5-31. Scenario 5.4 deployment diagram.

6 Complementary lab experiments

As already introduced in Section 3.2 (Methodology), a set of complementary lab experiments have also been performed too, as the name suggests, complement the work addressed in Demos #4 and #5, previously described. Specifically, these experiments are intended (i) to explore one of the quantifiable targets assigned to this WP6 in the Hexa-X work plan (QT 3d – Improvements on the Network Energy Efficiency – see Section 7.1.3.4) and (ii) to explore also some other topics that were considered interesting in the WP6 consortium, including the automated extreme-edge resources discovery mechanisms (closely related with the extreme-edge volatile resources orchestration), and the possible integration of the radio part in Scenario 5.1 (though the radio part is not in the scope of WP6, it was considered this scenario could be even more realistic taking in account the radio-related aspects, at least with a small-scale complementary experiment). In the following subsections, all these complementary lab experiments are described. Although, for the sake of simplicity, they are not treated in the same level of detail as the demos described above, they are considered to provide interesting complementary information to support the evaluation of the M&O mechanisms addressed in this document.

6.1 Network energy efficiency

The main motivation for this experiment is the validation of one of the Quantifiable Targets (QT) assigned to this WP6 in the Hexa-X project plan, specifically the QT "3d", regarding the improvement of the network energy efficiency using predictive orchestration. This QT is specifically evaluated in Section7.1.3.4, but the experiment that has been carried out to address that evaluation is described here.

The chosen experiment is based on a V2X scenario that is assumed to be deployed on the edge domain. As it is well-known, deploying resources at the edge of the network can provide shorter response times than those provided by a central cloud located farther away. This is particularly relevant for V2X scenarios, given the stringent delivery requirements of URLLC-type services. To efficiently provide these types of URLLC services, resources should be scaled up/down optimally as traffic load increases/decreases while guaranteeing the QoS. This is challenging due to the importance of being energy efficient. On the one hand, if all resources are active, the QoS is guaranteed, but wasting high levels of energy. On the other hand, if few resources are available, the system will be more energy-efficient but cause QoS disruption.

<u>Experiment description</u>

A V2X scenario is considered where vehicles send packets to a MEC server, e.g., Road-side Unit (RSU), to match URLLC-type service requirements. Although the scenario assumes a MEC deployment, it is not bound to any particular type of technology, provided that the server is relatively close to the users for latency considerations. The RSU can deploy as many servers as needed on demand to guarantee V2X services. Vehicles send packets at a certain rate, and these packets have a certain service time. Servers are modelled following an M/M/k queueing system [GDH+13]. Vehicles implement a Tele-operated Driving (ToD) application where part or all the tasks in the act of driving a vehicle are performed by a remote server. When an autonomous vehicle detects the need for remote support, it will share all the camera and sensor data (from RADAR or LIDAR sensors) to provide the server with adequate information about the environment. The server can then provide appropriate instructions to help the autonomous vehicle resolve the issues. For this purpose, the reliability and latency requirements to operate this service are 99.999% and 100ms, respectively [5GA21].

The energy consumption model is presented next. The edge deployment is modelled with carrier-grade servers, which consume 270W at their peak power and 150W while in idle mode [DPE]. The total power consumption is computed as the sum of (i) the power of having active servers (i.e., in idle mode); and (ii) the power associated with the server load. In this way, if an algorithm is capable of providing the same performance guarantees (reliability and latency) using fewer resources by accurately anticipating changes in demand (both to activate and to deactivate resources accordingly), it will result more efficient.

To simulate realistic road traffic in the experiment traces from Corso Agnelli Street in Torino (Italy) [MKV+22] on the last day of January 2022 have been used. This trace consists of traffic flow measurements reported each 5 min by a road probe. The 5-minute average number of vehicles is depicted in Figure 6-1, which corresponds to a typical workday pattern, with two periods of heavy traffic over the day.

This experiment consists of the simulation of a carrier-grade server's farm architecture with 150 servers, each one supporting up to 16 simultaneous requests. In this architecture, the following four scenarios were considered:

1. Peak-load dimensioning (i.e., no orchestration). This is the worst-case scenario, where all servers consume the maximum energy. In this case, all servers are active regardless of the traffic demand. This causes a complete waste of resources even though the QoS is guaranteed.
2. Oracle. In this case, the exact number of servers needed to match the QoS requirements are calculated. Therefore, the orchestrator accommodates the current traffic demand with a sufficient number of resources resulting in a more efficient approach. However, it is considered that this method is not realistic (or applicable in all cases) since knowing the current load can be challenging.
3. Predictive orchestration. In this case, the traffic load is predicted based on the load history. This implementation is intended to be more realistic than the previous one. This approach is based on using an LSTM Recurrent Neural Network (RNN) [YSH+19] [LST] to perform the load predictions.

The network energy efficiency model is evaluated by performing simulation experiments using a discrete event simulator, namely Ciw [PKH+19]. The simulations have been executed on an Intel(R) Core(TM) i7-1065G7 CPU @ 1.50 GHz based computer. Finally, the different KPIs considered in this experiment are described in Section 7.3.3.

Results

Results are depicted in Figure 6-2. For the **peak-load dimensioning** scenario, the minimum number of resources to accommodate the peak-load demand is calculated. This results in a 2551.90 W of power consumption. For the **oracle** scenario, the exact number of resources for each traffic demand is calculated. This yields a 158.73 W power consumption, resulting in a very effective approach in terms of power consumption. For the **predictive orchestration** scenario, the next traffic demand and scale-up resources are predicted. This results in a 324.769 W, getting very close to the optimal solution in the oracle scenario.

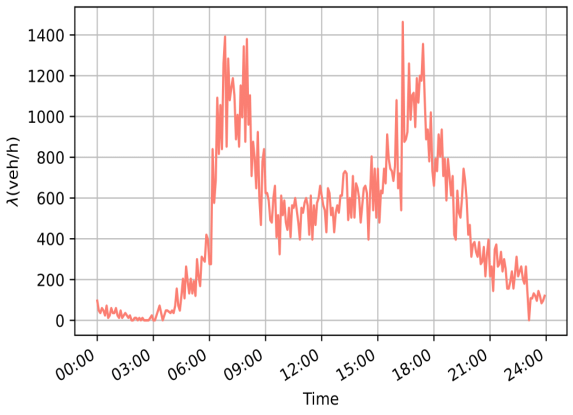


Figure 6-1. Vehicular rate at Corso Agnelli street in Torino during a day.

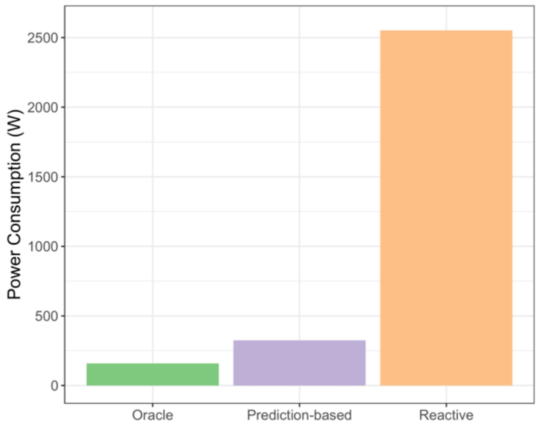


Figure 6-2. Power consumption results.

6.2 Extreme-edge nodes discovery

The dynamic discovery of extreme-edge nodes is a feature implemented in the resource orchestrator (operating at the infrastructure layer) developed for Scenario 5.1 (see Section 5.3.1). It allows keeping the resource inventory continuously aligned with the real-time availability of resources that can be used to run the services in the extreme-edge domains. The information provided by the resource inventory is used to feed the resource allocation decisions at the upper-layer service orchestrator (operating at the network and service layer) during the provisioning and runtime phases. This also enables the automated migration of application components in response to changes in the availability of volatile computing nodes at the extreme-edge, which can be considered a control loop action. As such, the discovery of extreme-edge nodes can have a role in the monitoring stage of a control loop. The information about nodes joining or leaving the infrastructure can be notified to components working at the network and/or service layers within the analysis and decision stages of the control loop, which can rely on AI/ML-based algorithms.

The resource discovery component is able to work on top of different kinds of clusters (Kubernetes, OpenStack, etc.), exploiting the abstraction layer that unifies the interaction with the various cloud platforms. For K8s-based scenarios, the discovery capability exploits the internal functionalities offered by K8s-like platforms to manage the nodes of the controlled clusters, exploiting their open APIs to watch over different clusters and detect when nodes join, leave or are marked as not available. More in detail, the resource discovery component acts as a K8s client, and it registers with the clusters in order to receive notifications about events related to their nodes. It should be noted that the current implementation targets a single administrative domain organized in multiple clusters, where each of them can be based on different technologies and rely on a specific type of edge platform. In this scenario, the resource inventory is maintained as a centralized component of the infrastructure layer, and it exposes APIs, with both query and subscribe/notify patterns, to enable the synchronization with upper-layer elements that are supposed to belong to the same administrator, with full visibility on the infrastructure topology. Extensions to support federation would require more advanced access control mechanisms at the inventory APIs, combined with procedures for abstracting, aggregating and regulating the exposure of information, also on a per-node basis, towards different administrative domains.

In order to test, validate and measure the performance of the node discovery feature, the K3d [K3D] tool is used to emulate a scenario with a K8s cluster composed of a master node and five worker nodes, each representing volatile extreme-edge resources (see Figure 6-3). K3d is a lightweight wrapper to run K3s in Docker, and it allows to creation of clusters on demand and add/remove worker nodes in a programmable way. In order to resemble the dynamicity of a real-world scenario where worker nodes can join/leave the master node, K3d was used to quickly spawn and delete worker nodes as Docker containers: nodes join the cluster and remain there for a given time interval, then leave the cluster for another time interval. Each extreme-edge node has been modelled with its own behaviour in terms of mobility and volatility, changing the frequency and the time intervals of the presence, as shown in Table 6-1.

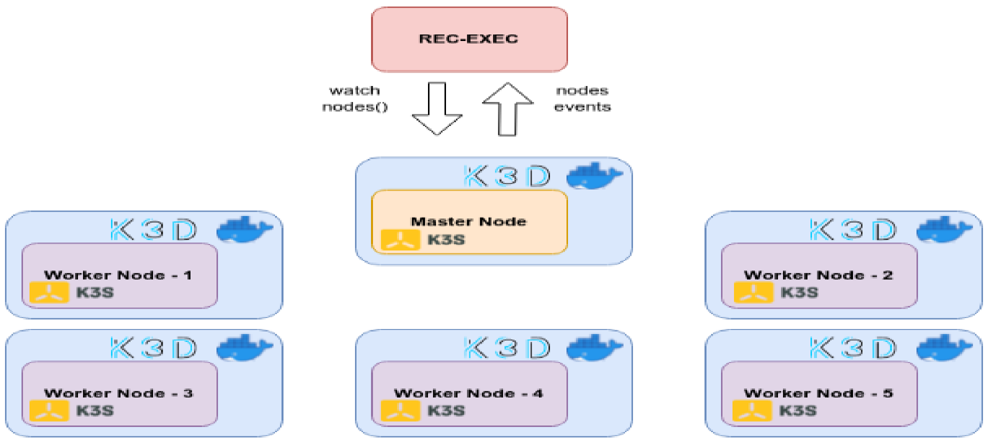


Figure 6-3. Node discovery test scenario.

In the current test, it is assumed nodes are joining and leaving the cluster at regular intervals, with cycles of 10 joining/leaving actions. Analysing the timestamps of the commands sent to K3d and the ones associated with the related update of the resource inventory, the statistics of the time intervals required by the system to synchronize with the nodes' events have been elaborated.

Table 6-1. Modelling of extreme-edge nodes' volatility.

Extreme-edge node	Time in the cluster	Time out of the cluster
Worker-1	30 s	5 s
Worker-2	35 s	10 s
Worker-3	40 s	15 s
Worker-4	45 s	20 s
Worker-5	50 s	25 s

Results

It has been measured that the average time needed to spawn a node with K3d, including the time to discover the new node itself, is 4.9 s, with a maximum of 6.7 s and a standard deviation of 0,46. The leaving time is 1.4 seconds on average, with a maximum of 2.7s and a standard deviation of

0.39 (see Table 6-2). The additional overhead required to synchronize the resource inventory is negligible (in the order of milliseconds).

Table 6-2. Synchronisation time for nodes' joining and leaving actions.

Event	Average time	Max time	Std Deviation
Node joining the cluster	4.9s	6.7s	0.46
Node leaving the cluster	1.4s	2.7s	0.39

Finally, assuming to switch off the nodes without sending the K3d leave command and relying entirely on K3s internal procedures to detect the nodes' unavailability (marking them as Not Ready), the time interval for the leaving detection mostly depends on the internal timers set for the nodes' heartbeats in K3s (the 40 s in the default configuration). These timers are configurable, with the default values set to achieve a good trade-off between accuracy, the load of the heartbeat traffic, and system stability. Reducing the timers increases the number of notifications about the changes in nodes' status, and it may lead to some inconsistency and unstable conditions, especially in case of poor connectivity between the worker and the master nodes, even if the workers are alive and correctly running since they tend to lose the connectivity with the master. It should be noted that the information related to the missing availability of nodes can be used at the upper-layer service orchestration to trigger migration actions, and, in case of poor accuracy, they may lead to unnecessary delays or breaks in the service execution and continuity. The internal migration strategy implemented by K8s/K3s starts to monitor the nodes declared as "Not Ready" and, if they do not become available within 5 minutes (configurable down to 20 seconds), automatically triggers a migration action, whose duration depends on the characteristics of the containers. As an alternative, the migration can be initiated by the service orchestrator operating at the upper layer or specifying some parameters in the application description. However, the overall system logic should be able to guarantee maximum service continuity, using service migration only when extremely necessary. In this sense, it becomes important to have a realistic and updated view of the status of the various nodes where the service is running. A possible solution to improve the accuracy of extreme-edge nodes' discovery and monitoring is the adoption of an adaptive heartbeat mechanism which is dynamically configured to better match the behaviour and the profile of the various nodes, e.g., adjusting the heartbeat timers and the availability decisions with per-node criteria, on the basis of the nature and behaviour of each node.

6.3 Simu5G in Scenario 5.1

The evaluation of the impact that the B5G/6G RAN might have on Scenario 5.1 is of paramount importance in order to clarify future scenario enhancements aiming at achieving a higher TRL and full integration with the B5G/6G mobile networks stack, i.e., adding a real RAN to the scenario, implementing it on a real-life scenario, etc. This experiment was created as first-step research towards these objectives. As explained in Section 5.3.2.2, Simu5G [NSS+20] allows the creation of a wide range of network topologies that include UPFs, PGWs, gNBs and UEs components as desired by the simulated network designer.

The UE and gNB components and the communications between them are modelled as OMNeT++ components, and, consequently, they can be parametrised as preferred in order to fit in a given scenario. Within the scope of this experiment, Simu5G is used to simulate a realistic implementation of a 5G transport network within the building blocks that comprise Scenario 5.1 (see Section 5.3.1) and to study the delays that emerge due to the integration of the transport network, and its potential impact on the Scenario components. It is important to remark that the *Sumo Extreme-edge* and *Traffic Lights Control Logic* components (see Figure 5-4) work on the extreme-edge domain; thus, they are constrained by low delay requirements in order to be able to act as real-time components.

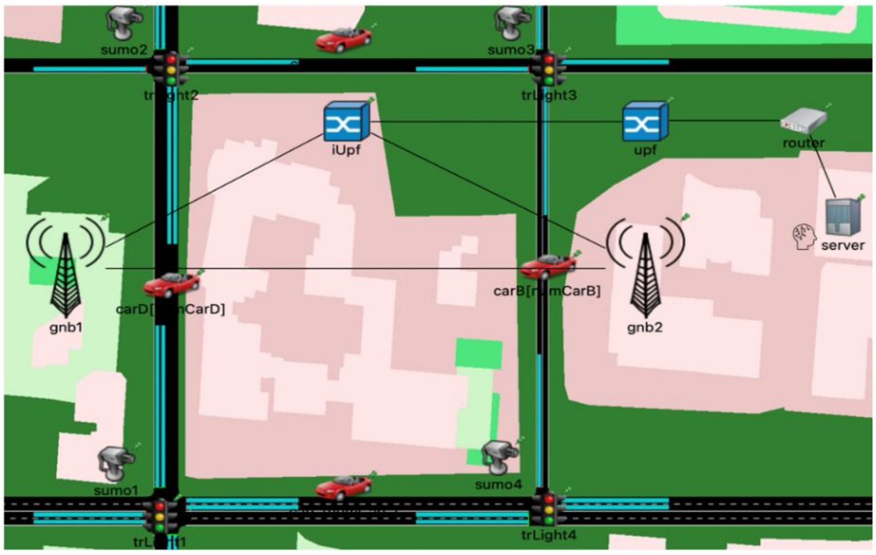


Figure 6-4. Simu5G mapping to Scenario 5.1 software components.

Simu5G simulated scenario

As depicted in Figure 6-4, all the components from Scenario 5.1 (i.e., the *Sumo Extreme-edge* instances, the *Traffic Lights Control Logic* instances and the *Reinforcement Learning Agent* instance) have been mapped to modelled pairs on Simu5G, in such a way that even the domain placement of those resources has been replicated on Simu5G.

Table 6-3. Scenario 5.1 components modelling[11].

Component	Rate [msg/s]		Size [bytes]		Inter-packet-time [ms]	
	Min	Max	Min	Max	Min	Max
sumoee1	63.80	90.70	52	1450	11.03	15.67
sumoee2	55.00	74.10	52	1450	13.50	18.18
sumoee3	56.40	80.20	52	1450	12.47	17.73
sumoee4	63.40	88.60	52	1450	11.29	15.77
ai_agent	41.20	125.80	52	364	7.95	24.27
tl_ctrl_1	1.00	30.20	52	282	33.11	1000
tl_ctrl_2	1.00	28.90	52	260	1000	34.60
tl_ctrl_3	1.00	29.10	52	254	1000	34.36
tl_ctrl_4	1.00	29.30	52	261	1000	34.13

As it can be seen, the *Sumo Extreme-edge* and the *Traffic Lights Control Logic* instances are located at the extreme-edge domain, while the *Reinforcement Learning Agent* component has been allocated on a server behind the UPF at the edge domain. To model each of the aforementioned components, a delay profile that replicates their behaviour on Scenario 5.1 has been generated per component. Table 6-3 reflects the network behaviour of each component instance from a message rate and message size perspective. To obtain a more realistic configuration, the simulated network is loaded with a variable number of vehicles deployed as UEs within the network. Said UEs represent the active users in the scenario, i.e., those who are

[11] '*sumoee1*' to '*sumoee4*' represent the four *Sumo Extreme-edge* component instances. '*ai_agent*' is the *Reinforcement Learning Agent* component instance. '*tl_ctrl_1*' to '*tl_ctrl_4*' represent the four *Traffic Lights Control Logic* component instances.

downloading or uploading data using the network, thus actively using it. More in detail, each user is consuming a video-streaming service, which continuously transmits a video in 720p format, having a bit rate of 2.4 Mbps and a frame rate of 25fps. The video-streaming service is modelled in the simulator as a UDP application, which transmits packets of variable size every 40ms. The size of each packet is chosen randomly from a uniform distribution, between 22593 and 25406 bytes, to follow the behaviour of a realistic traffic trace. Each experiment is repeated three times to achieve statistical soundness. Confidence intervals at 95% are reported when visible.

Table 6-4. Scenario 5.1 main simulation parameters.

Parameter Name	Value
Carrier frequency	2 GHz
System Bandwidth	50MHz
gNB Tx Power	46 dBm
gNB antenna gain	8 dBi
gNB noise figure	5 dB
UE antenna gain	0 dBi
UE noise figure	7 dB
CQI reporting period	40 TTIs
Path loss model	[TR873]
UE mobility	Linear Mobility
UE speed	U[36,72] km/h
Background Traffic type	Video streaming

<u>Experiment Results</u>

First of all, the impact of Scenario 5.1 traffic on a B5G/6G RAN is analysed. Figure 6-5 shows the average number of resource blocks consumed in uplink and downlink, respectively, by the traffic of Scenario 5.1.

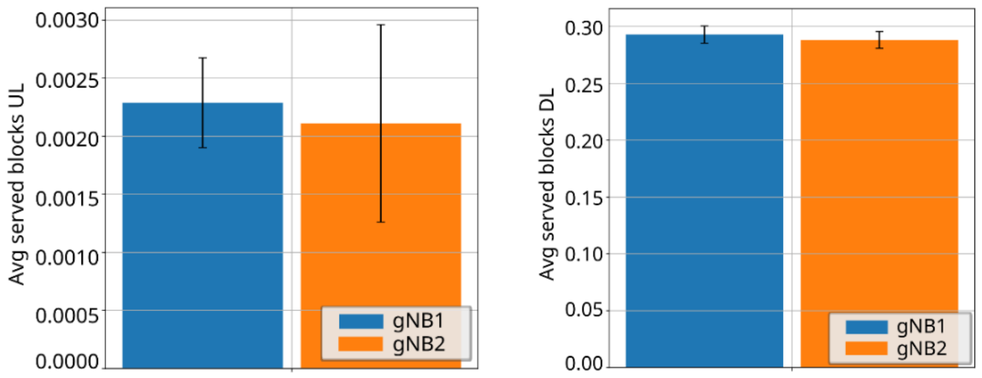


Figure 6-5. Average number of resource blocks in Uplink (left) and Downlink (right)

The two plots show that the impact on network resources is very low in both cases, having a slightly larger impact in the downlink direction as compared to the uplink direction. The former direction is indeed carrying the traffic from the *Reinforcement Learning Agent* towards the traffic lights, which can occur at more frequent timings (~8ms), thus generating more traffic. Moreover, it also carries the downlink leg of the traffic between the traffic lights and the sumo controllers. Figure 6-6, Figure 6-7, and Figure 6-8 show the performance of the traffic management service from the perspective of its three service components, i.e., the *RL Agent*, the *Traffic Lights Control Logic* and the *SUMO Extreme-edge component*. Note that the aforementioned traffic is generated respectively by the three mentioned components. Three different experiments have been carried out, considering an increasing number of background traffic sources, namely 0 (no background traffic), 5 and 10 users. As it can be seen from the three plots, the service delay is always below 16ms, even for the higher traffic loads, thus confirming the feasibility of the proposed methodology.

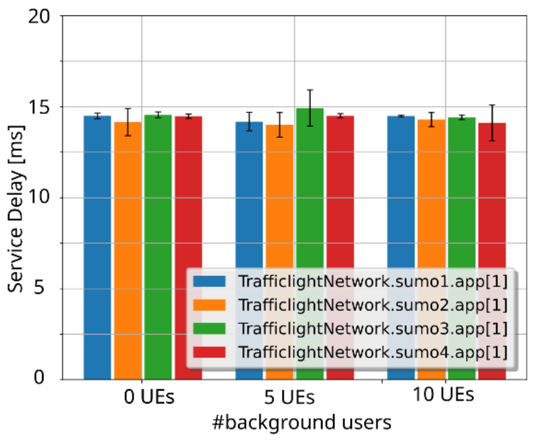


Figure 6-6. Average delay of the communication between the *SUMO Extreme-edge* components and the *RL Agent*.

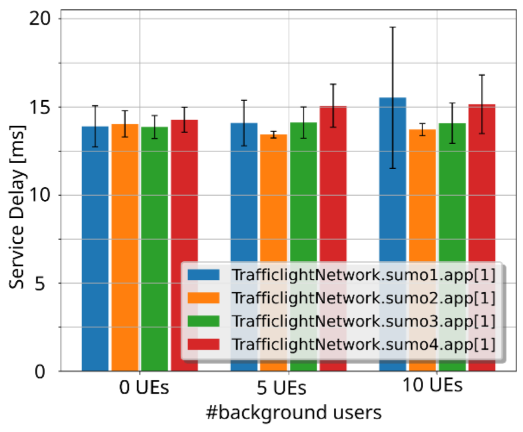


Figure 6-7. Average delay of the communication between the *Traffic Lights Control Logic* and the *SUMO Ext. Edge*.

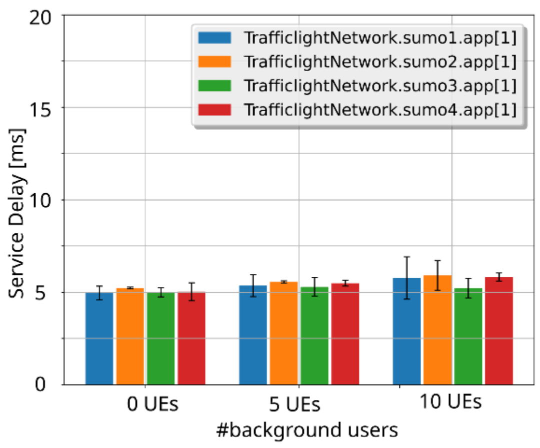


Figure 6-8. Average delay of the communication between the *RL Agent* and the *Traffic Lights Control Logic* components.

7 Evaluation

This section presents the "Evaluation" part of the document, which includes the WP6 contribution to the Hexa-X objectives (Section 7.1), which hence describes the main WP6 outputs towards the objective in scope in this deliverable (Objective 3, in Section 7.1.1), the main measurable results (Section 7.1.2) and the results regarding the WP6 quantifiable targets (Section 7.1.3). The evaluation also considers the validation of the Hexa-X M&O architectural design, which was the main outcome of the previous Deliverable D6.2 [HEX22-D62], considering how this architectural design has been applied to implement both Demos #4 and #5 (Section 7.2). Also, in line with the work from D6.2, the evaluation also considers the main KPIs, KVIs and Core Capabilities defined in that document, and that have been considered in the scope of the demos and the lab experiments presented in this document (Section 7.3). Finally, the evaluation also considers the main lesson learnt (Section 7.4) and some hints for future work (Section 7.5) regarding the work performed in this WP6.

7.1 WP6 contribution to the Hexa-X objectives

The Hexa-X project has defined a number of objectives in its work programme, from which two of them were linked to this WP6. They are the following:

- Objective 1 – Foundations for an end-to-end system towards 6G, aimed to build a vision and roadmap for the B5G/6G end-to-end system.
- Objective 3 – Connecting intelligence towards 6G, which aims to turn AI/ML to an essential component of B5G/6G technology[12].

To address these overall objectives different WP6-specific objectives were also defined in the Hexa-X work programme, namely:

- Targeting Objective 1:
    - **WPO6.1**: Identification and selection of disruptive trends and technologies, and gap analysis of resource description, service management and orchestration towards future orchestrators.
- Targeting Objectives 1 and 3:
    - **WPO6.2**: Provide necessary means for the automation and network programmability of B5G/6G infrastructures, to address the heterogeneity of service requirements, the extended complexity of the infrastructure and the need for utmost network efficiency in a sustainable network (service creation time, amount of used resources, reliability and network dynamicity with massive amount of network functions) without neglecting the performance, scalability, and resiliency of the network functions.
    - **WPO6.3**: Provide intent-based mechanisms for elaborating on requirements, diagnosing the performance of networks and services, modelling/abstracting services/networks, as well as implementing corrective actions through CI/CD.
- Targeting Objective 3:
    - **WPO6.4**: Support orchestration of a wide variety of service definitions and decompositions, including (traditional) virtual appliances, microservices and containers, and serverless functions in all domains.
    - **WPO6.5**: Design and evaluate efficient cognitive-based service management and orchestration mechanisms based on optimised placement, resource optimisation and dynamic allocation.
    - **WPO6.6**: Demonstrate algorithms for data-driven device-edge-cloud continuum management.

Objective WPO6.1 was addressed in the WP6 Task 6.1, being the main outcome the initial WP6 Deliverable D6.1 [HEX21-D61], which describes the “gaps, features and enablers for B5G/6G service management and orchestration”. On the other hand, objectives from WPO6.2 to WPO6.4 are in fact the objectives of the previous Deliverable D6.2 itself, targeting the design of the service M&O functionalities [HEX22-D62] (see Section 3.1 in that document). The fulfilment of these objectives is reported in such D6.2, being summarized in Section 12 (Conclusions) in that document. WPO6.5 is also partially addressed in D6.2, in what regards the “design” part, since that deliverable provides, as a whole, the Hexa-X M&O architectural design itself. However, the “evaluation” part, as well as the whole WPO6.6 (targeting demonstrations) are addressed right in this Deliverable D6.3 you are reading now, as part of the “final evaluation of service management and orchestration mechanisms” (the main topic of this document).

As it can be seen in the previous bullets list, both WPO6.5 and WPO6.6 are targeting the overall Hexa-X Objective 3, so in the following subsections we will describe how that Objective 3 has been fulfilled, in what regards these WPO6.5 and WPO6.6 objectives, i.e., by means of the demonstration activities described in the previous sections 4, 5, and 6 in this document.

[12] Other Work Packages in the project are also addressing these (and other) objectives from their respective work scopes.

7.1.1 WP6 output towards Objective 3

As mentioned, Objective 3, as a whole, targets "connecting intelligence towards 6G", aiming to turn AI/ML to an essential component of B5G/6G technology. More specifically, the objective targets three aspects: (i) the role that AI will have in transforming the conventional air-interface design; (ii) the methods and algorithms for ensuring that AI, at infrastructure or service level, will be secure/ trustworthy, sustainable (e.g., operating with energy-efficiency), explainable and efficient, with respect to resource consumption and performance delivered; (iii) AI-powered means for enhancing the orchestration operations, and, ultimately, for empowering enhanced mobile services. From these three aspects, WP6 addresses of course item (iii) in what regards enhancing orchestration operations, and partially item (ii) also, in what regards security/trustworthiness and sustainability, also in liaison with M&O aspects.

As commented, part of these aspects, in what regards the evaluation of the state-of-the-art and the design activities, have been already addressed in [HEX21-D61] and [HEX22-D62], while the demonstration and the evaluation part rely on this deliverable, targeting the evaluation of "cognitive-based service M&O mechanisms based on optimised placement, resource optimisation and dynamic allocation" (WPO6.5) and the demonstration of "algorithms for data-driven device-edge-cloud continuum management" (WPO6.6).

Specifically, cognitive-based service M&O mechanisms based on optimised placement, resource optimisation and dynamic allocation have been addressed in both, Demo #4 and Demo #5. In Demo #4 this has been specifically addressed in Scenario 4.2, which showcases how AI/ML can be used to improve resource optimization, targeting anomaly detection and performance degradation, based on the optimised network functions placement along with increased automation and programmability (see Section 4.3.2). On the other hand Demo #5 also addresses this objective in Scenario 5.2 (Prediction-based URLLC service orchestration and optimization), which aims at demonstrating the usage of AI/ML algorithms to anticipate the resource needs of the network, and pre-emptively activate the related services to avoid delays in using an URLLC application. In this case the AI/ML-driven resource optimization function is used to proactively activate or deactivate new resources or re-route network traffic accordingly (see Section 5.3.2).

Regarding the "algorithms for data-driven device-edge-cloud continuum management" (WPO6.6) this is also addressed in both, Demo #4 and Demo #5. Demo #4 addresses the "device-edge-cloud continuum management" topic in Scenario 4.1 (see Section 4.3.1), which is specifically devoted to this. Also, since this Scenario 4.1 is the basis on which Scenarios 4.2 and 4.3 rely, and since both Scenarios 4.1 and 4.2 are based on using AI/ML techniques (for the functions placement and to perform predictive orchestration – see Sections 4.3.2 and 4.3.3), it can be stated that objective WPO6.6 is fully addressed by means of this Demo #4. On the other hand, Demo #5 also address specifically the "device-edge-cloud continuum management" using a data-driven approach in Scenario 5.2 (see Section 5.3.2), already mentioned above, since the prediction-based orchestration actions in that scenario are performed considering the resources in the continuum from the devices (extreme-edge) up to the cloud.

However, besides those specific topics mentioned in objectives WPO6.5 and WPO6.6 (which are considered fulfilled as described above), we consider the demonstration activities performed in this WP6 go even beyond, in the aim of actually "turning AI/ML to an essential component of the B5G/6G technology", as stated in the overall Hexa-X Objective 3 definition. For instance, AI/ML is applied also in Demo #5 – Scenario 5.1 (Continuum orchestration of AI/ML-driven Traffic Lights Control Service – Section 5.3.1). Although in this case the AI/ML components are not part of the M&O processes themselves (just part of the road-traffic managed service), we consider it is a good example on how AI/ML could be used in future 6G networks. Also, Scenario 5.4 (MLOps, in Section 5.3.4) addresses the challenge of developing and deploying AI/ML components on the MNO scope, which can be considered of course something essential to make AI/ML part of the future 6G technology. Finally, AI/ML is also considered in one of the complementary lab experiments in Section 6, specifically the one evaluating the usage of AI/ML techniques to improve the network energy efficiency (Section 6.1).

7.1.2 WP6 measurable results towards Objective 3

According to the Hexa-X workplan, measurable results linked to Objective 3 are:

a) Designs for data-driven wireless transceiver of low complexity, either "block-per-block", or, by means of "end-to-end" optimisation.
b) Frameworks for data-centric hardware impairment mitigation and adaptivity to the wireless environment.
c) Concepts and mechanisms to support and manage collaborative AI components across the network, also leveraging Federated Learning (FL) and deployment of eXplainable AI (XAI) models.
d) Development and assessment of intelligent orchestration methods, such as predictive orchestration.
e) Implementation of continuum management of device, edge, RAN, and cloud.
f) Development of interfaces and abstractions to increase the full network programmability and E2E seamless integration management mechanisms that includes data-driven optimisation and adaptative monitoring.

As it can be appreciated some of these measurable results are out of scope for this WP6, namely (a) and (b), since they address hardware related topics which are in the scope of other WPs (as known, WP6 focuses on M&O). Item (c) partially matches the WP6 scope, in what regards the overall statement about the "concepts and mechanisms to support and *manage* collaborative AI components across the network", though the specific FL and XAI technologies are out of scope in our case, since they are specifically addressed in other WPs (WP4 and WP5). Obviously, items (d), (e) and (f) fully match the WP6 scope. In the following paragraphs we describe compliance with these measurable results in what regards WP6, i.e., regarding items (c), (d), (e) and (f) in the Objective 3 definition.

Item (c): Concepts and mechanisms to support and manage collaborative AI components across the network.

The fulfilment of this measurable result is in the scope of the previous Deliverable D6.2 [HEX22-D62], addressing the M&O architectural design of the novel orchestration and management mechanisms for Hexa-X. Specifically, the support for "the management of collaborative AI components across the network" was included in that deliverable as a functional requirement for the M&O architectural design itself (see Section 5.2.2 – Item 5 in [HEX22-D62]) and, in fact, such collaborative AI components are represented in the provided architectural design (Section 6 in [HEX22-D62]). In summary, from the M&O perspective, those collaborative AI components (that could be FL-based, XAI-based, or relying on any other technology) would be just a specific kind of "managed objects" (Section 6.1 in [HEX22-D62]), which could be managed using the M&O mechanisms described in Section 7 (also in [HEX22-D62]). M&O actions could be also supported by specific AI/ML Functions in the Network Layer (Figure 6-1 in [HEX22-D62]).

Item (d): Development and assessment of intelligent orchestration methods, such as predictive orchestration.

This item was also addressed in the previous [HEX22-D62], as part of the M&O architectural design, although in this case some of these "intelligent orchestration methods" (such as predictive orchestration) have been demonstrated also in Demo #4 (Scenarios 4.2 and 4.3) and Demo #5 (Scenario 5.2), as it has been already explained in the previous Section 7.1.1. The mapping of these practical demonstrations with the architectural design in [HEX22-D62] basically consists on implementing the intelligent orchestration methods as part of the AI/ML Functions block, which support the Management Functions block itself (see Figure 6-1 in [HEX22-D62]).

Item (e): Implementation of continuum management of device, edge, RAN, and cloud.

Again, this is one of the main innovation topics addressed in [HEX22-D62] (see Section 5.3 in that document), targeting the integration of the extreme-edge domain as an additional set of infrastructure resources from the M&O perspective. In this case, this has been also one of the main work items in Demos #4 and #5, and also in one of the complementary lab experiments in Section 6, specifically the one devoted to the extreme-edge nodes discovery (Section 6.2). The integration of the RAN has been also explored in the experiment regarding using Simu5G in the Demo #5 Scenario 5.1 (Section 6.3).

Item (f): Development of interfaces and abstractions to increase the full network programmability and E2E seamless integration management mechanisms that includes data-driven optimisation and adaptative monitoring.

This topic was also addressed in the previous Deliverable D6.2 [HEX22-D62] as part of the M&O architectural design provided in that document, and specifically, in what regards the API Management Exposure concept (Section 6.2.3 in such document), which represents the functional block enabling and regulating communication among the different M&O resources, within and across administrative domains, making possible the full network programmability and the E2E seamless integration management mechanisms. Also, beyond D6.2, a small-scale implementation of this API Management Exposure concept has been performed in Demo #5 – Scenario #4 (MLOps) to enable the communication between the simulated SW Vendor (the stakeholder providing the AI/ML models) and the MNO (Section 5.3.4).

7.1.3 WP6 quantifiable targets towards Objective 3

Following the Hexa-X workplan, Objective 3 requires to verify eight Quantifiable Targets (QT), namely:

- **QT 3a**: Network reconfiguration (creation, composition and scaling times) to be performed by (>10%) of the prediction horizon.
- **QT 3b**: Improvement by (>90%) in time to onboard new resources from other domains and manage the addition/removal of elements from the network.
- **QT 3c**: Increase the service continuity by reducing the downtime by (>80%).
- **QT 3d**: Increase network energy efficiency by (>50%) applying predictive orchestration.
- **QT 3e**: Increased AI algorithm robustness to system parameter volatility; significant Bit Error Rate (BER)/Block-Error Rate (BLER) gain, as compared to classical approaches.
- **QT 3f**: Number of dynamically collaborating AI components in the network (>1000).
- **QT 3g**: The accuracy of an XAI model within (<10%) of "black box" solutions (e.g., Deep Neural Networks DNNs).
- **QT 3h**: Energy reduction of a factor of (>10) at the infrastructure level and a factor of (>100) at the user devices' side, as a result of AI-based workload offloading.

Of these, WP6 has been responsible for verifying the four of them with a clear relationship with the M&O topic (i.e., those from QT 3a to QT 3d), while the other four (from QT 3e to QT 3h) have been addressed in other WPs.

However, it has to be stated that, although these WP6-related QTs are defined with specific numerical values, it was found that their precise validation was not entirely possible. The reason for this is the lack of a clear baseline in the QTs definition. In other words, although specific improvement percentages are defined for the different aspects mentioned in each QT, it is not clearly defined "with respect to what" these improvement percentages are expected to be applied. I.e., it cannot be stated "in a general way" that certain KPIs will be improved in a specific percentage if the technology benchmark is not clearly stated. However, in order to address the defined QTs, it was considered that anyway, it could be demonstrated that the requested improvements could be achieved at least in particular cases, and that, although these particular cases cannot be generalised, they can at least give an idea about whether the targets could be actually met. This is the approach that has been taken to address the four QTs mentioned above, defining specific particular testing cases for each of them. In some cases, these tests have been performed in the context of Demos #4 and #5, while in other cases, some of the lab experiments described in Section 6 have been used for that.

It is worth noting that this procedure does not allow claiming QT fulfilment in general, i.e., with respect to any orchestration system or any kind of orchestrated network service; however, this procedure at least gives an idea of whether the proposed targets can actually be achievable.

According to this approach, the following subsections describe the work performed for each of the quantifiable targets, together with the corresponding results.

7.1.3.1 QT 3a: Improvement in network reconfiguration times

This QT requests to validate the feasibility of achieving network reconfiguration (regarding creation, composition, and scaling times) to be performed by (>10%) on the prediction horizon. From the WP6 standpoint, this objective relates to making predictions (e.g., using data-driven AI/ML-enabled mechanisms) to perform the mentioned network reconfiguration actions (creation, composition, and scaling regarding complete services), providing a 10% improvement in terms of time. Also, the understanding is that the baseline should be those scenarios where predictive mechanisms are not used. As previously mentioned, it is considered not feasible to validate this QT in a general way, i.e., it should be hard to claim that applying predictive orchestration could "always" produce *per se* a 10% gain in time regarding the mentioned network reconfiguration mechanisms without considering specific services, specific M&O platforms, or the specific network resources in scope. So, following the approach explained above, specific testing cases have been selected to evaluate this QT. Specifically, two approaches have been considered for this case: one of them evaluates the work performed in Demo #4 (since this demo addresses the usage of predictive algorithms), while the second one considers the work also performed in Scenario 5.2 (that also focuses on predictive orchestration). The following describes the approach and results for each of these approaches.

<u>Validation in Demo #4</u>

Regarding Demo #4, the target set by QT3a is interpreted as being able to perform predictive orchestration actions at least 10% of the prediction horizon earlier than a predicted event. Specifically, as mentioned in Section 7.1.1, the prediction horizon, based on the model characteristics and accuracy, is set at 8-10 minutes. This prediction horizon leads to an acceptable time window for action, according to QT3a, of at least 48-60 seconds before an event is predicted to occur. Additionally, the workflow time, which is the time needed to complete all the orchestration actions necessary for the reconfiguration, should also be reduced by more than 10%. In this demo, there are three scenarios, the first of which will be the base on which the other two will be used to validate the proposed M&O mechanisms. Namely, the "Cloud – Edge – Extreme-edge Continuum Orchestration", which allows the other two mechanisms, the "Function Placement" and the "Predictive Orchestration", to showcase the enhancements they provide to the M&O operations. The validation goals of this demo are to showcase how these mechanisms enhance M&O operations by improving the operation times and reducing service downtime in case of unexpected events. For the validation of these mechanisms, three versions of the M&O workflows are used, one for each scenario:

- typical M&O workflow (notification, action);
- reactive orchestration with placement optimization (using performance diagnosis and Function Placement);
- predictive orchestration (using performance/status prediction and reactive orchestration as a fallback).

The first workflow is used as the baseline against which the improvements from the other two are quantified and validated. As such, even though a lot of options exist to configure and fine-tune the existing M&O components, like the K8s controller manager, the default configurations are used for uniformity of expected results across different clusters. These three workflows are used to handle four types of events that can occur during automated operations in the industrial context of the demo. For each of these types, ten instances of "unexpected" behaviour are triggered manually, following the typical operational patterns of the industrial automation service. These four events are:

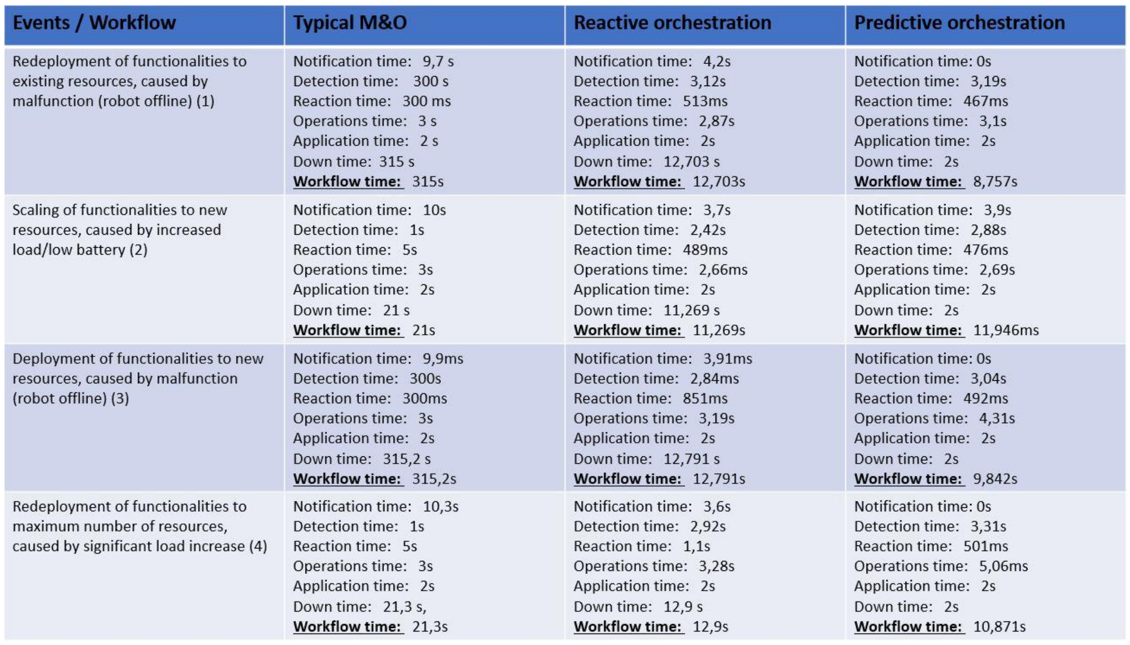
1) Redeployment of functionalities to existing resources caused by robot malfunction.
2) Scaling of functionalities to new resources caused by increasing load/low battery.
3) Deployment of functionalities to new resources caused by robot malfunction.

4) Redeployment of functionalities to the maximum number of resources caused by significant load increase.

For each of these types of events, the following time periods were measured:

- Notification time: the time it takes for the monitoring system to check for the status of the component/node.
- Detection time: the time it takes for the monitoring system to detect that there is an issue (including timeouts, retries, etc.).
- Reaction time: the time it takes for the corrective actions to be triggered on the respective components.
- Operations time: the time it takes for the corrective actions to be completed (functionality reallocation, scaling by commissioning resources, etc.).
- Application time: the time it takes for the service to be restored (mostly due to service initialization or management operations, in case it became unavailable)
- Downtime: the time that the service was unavailable.
- Workflow time: the time it takes from the unexpected event appearance until the service is available again.

In Figure 7-1 the averages of the collected time measurements from the 40 injected events are displayed in detail.



Events / Workflow	Typical M&O	Reactive orchestration	Predictive orchestration
Redeployment of functionalities to existing resources, caused by malfunction (robot offline) (1)	Notification time: 9,7 s Detection time: 300 s Reaction time: 300 ms Operations time: 3 s Application time: 2 s Down time: 315 s **Workflow time:** 315s	Notification time: 4,2s Detection time: 3,12s Reaction time: 513ms Operations time: 2,87s Application time: 2s Down time: 12,703 s **Workflow time:** 12,703s	Notification time: 0s Detection time: 3,19s Reaction time: 467ms Operations time: 3,1s Application time: 2s Down time: 2s **Workflow time:** 8,757s
Scaling of functionalities to new resources, caused by increased load/low battery (2)	Notification time: 10s Detection time: 1s Reaction time: 5s Operations time: 3s Application time: 2s Down time: 21 s **Workflow time:** 21s	Notification time: 3,7s Detection time: 2,42s Reaction time: 489ms Operations time: 2,66ms Application time: 2s Down time: 11,269 s **Workflow time:** 11,269s	Notification time: 3,9s Detection time: 2,88s Reaction time: 476ms Operations time: 2,69s Application time: 2s Down time: 2s **Workflow time:** 11,946ms
Deployment of functionalities to new resources, caused by malfunction (robot offline) (3)	Notification time: 9,9ms Detection time: 300s Reaction time: 300ms Operations time: 3s Application time: 2s Down time: 315,2 s **Workflow time:** 315,2s	Notification time: 3,91ms Detection time: 2,84ms Reaction time: 851ms Operations time: 3,19s Application time: 2s Down time: 12,791 s **Workflow time:** 12,791s	Notification time: 0s Detection time: 3,04s Reaction time: 492ms Operations time: 4,31s Application time: 2s Down time: 2s **Workflow time:** 9,842s
Redeployment of functionalities to maximum number of resources, caused by significant load increase (4)	Notification time: 10,3s Detection time: 1s Reaction time: 5s Operations time: 3s Application time: 2s Down time: 21,3 s, **Workflow time:** 21,3s	Notification time: 3,6s Detection time: 2,92s Reaction time: 1,1s Operations time: 3,28s Application time: 2s Down time: 12,9 s **Workflow time:** 12,9s	Notification time: 0s Detection time: 3,31s Reaction time: 501ms Operations time: 5,06ms Application time: 2s Down time: 2s **Workflow time:** 10,871s

Figure 7-1. Collected time measurements during unexpected events.

The noticeable difference between the type of events is that for events (1) and (3), the robot went offline due to malfunction, which sets in motion the K8s workflow to identify, wait for a response and finally evict the unavailable pods from that node to an available one. This uses, by default, a timeout of 300 seconds to prevent unnecessary eviction of pods due to random short-lived network unavailability. Also, for these two events (1) and (3), for the first M&O workflow, it is assumed that the additional required nodes and compute resources are already available in the cluster due to a lack of automated commissioning. For the Reactive and Predictive orchestration workflows, the dynamic commissioning of resources from already available resources is handled by the intelligent orchestration set of functionalities so as to optimize energy efficiency and load distribution. In Figure 7-2**,** the average duration of each workflow is shown for each event type.

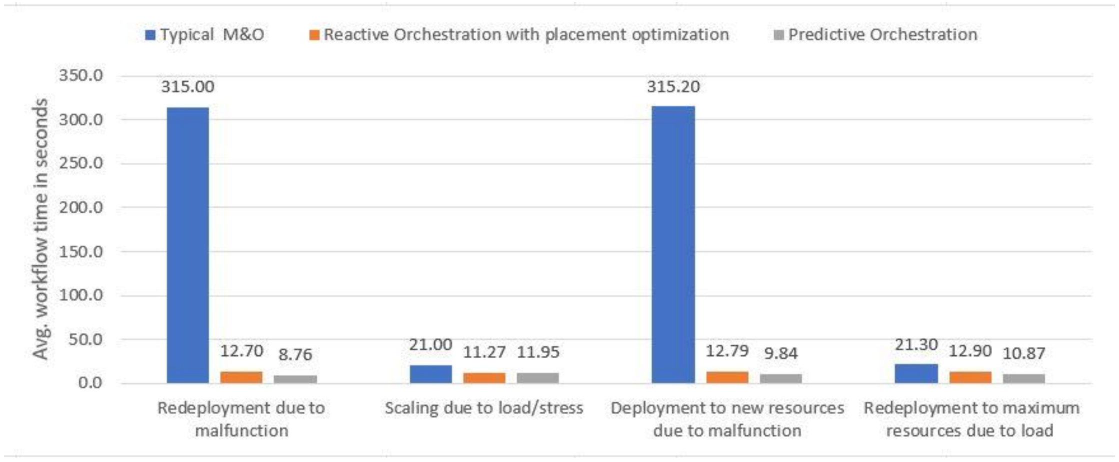


Figure 7-2. Average workflow time for each event type.

For the first two workflows, which are primarily reactive, the most time-consuming operation is the notification and detection steps. The monitoring system polls the infrastructure and services at various intervals, and information from multiple sources needs to be collected usually before identifying an issue. In contrast, the predictive orchestration workflow requires no notification step since the future values are predicted beforehand, thus moving on to the identification step immediately. The other steps require about the same time for action triggering and operations. For the final step, the application time is intrinsic to the service operations and is constant on the application's start-up, in case of a single instance, every time the service is migrated. While there is not a significant difference in the operations time, the main benefits from the introduction of the proposed mechanisms become apparent when we examine the service downtime. For the first two reactive workflows, the service is down for the complete duration of the operations exactly because these actions are triggered after the disruptive event has occurred. For the predictive workflow, on the other hand, the service is unavailable only for the time required for the actual service to initialize, as shown in Figure 7-3, meaning that this time could be close to or equal to zero, depending on the service.

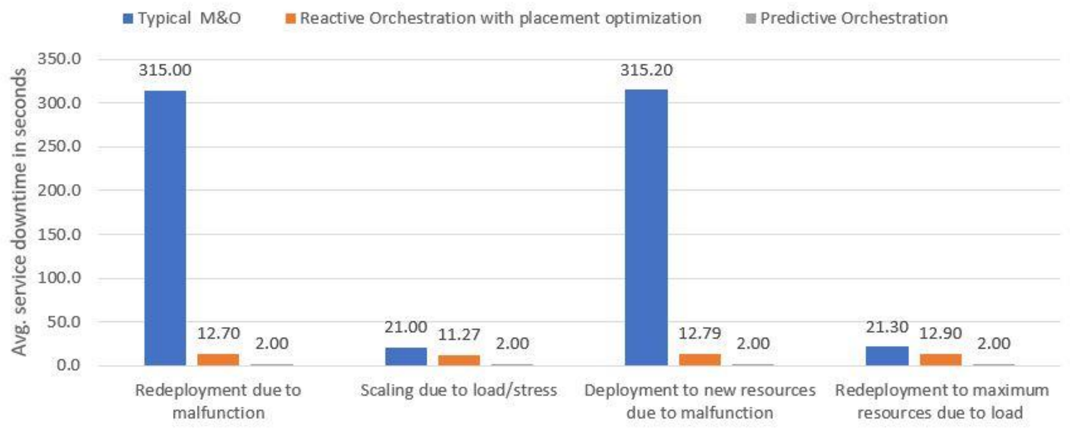


Figure 7-3. Average service downtime for each event type.

This improvement relies on the successful prediction of these "unexpected" events. To accomplish that, a kind of periodicity in these events is needed in order to be able to predict the future state of the infrastructure and services in a sufficient prediction horizon. Based on the examined services and tasks taking place in this industrial context and their time duration and relative periodicity, the prediction horizon was set at 8 – 10 minutes with the achieved prediction accuracy of around 80%. To fulfil the quantified target, i.e., to apply the orchestration actions at least 10% of the predicted time window before these events are predicted to happen, this would be at least 48-60 seconds earlier. This time limit is used as a constraint to signal the end of the time period in which a predicted event is deemed “possible”. While the status of the examined event is deemed “possible”, but the limit is not yet reached, the accuracy of the predictions is still verified by the latest measurements. When these new measurements validate the predictions enough, the predicted value error is minimal, while still in the acceptable time window, the predicted event is deemed as “sure”, and orchestrations actions are triggered to prevent it. If the predicted values fail to validate the event, then the predicted event is ignored, the predicted values are discarded, and a new prediction round begins. In that way, in the best case, the event is prevented successfully, and in the worst case, the event is considered to be not possible and is ignored. In the case that event finally occurs, it is subsequently be handled by the reactive orchestration mechanism as in the first two workflows.

The previously described and validated predictive orchestration approach focuses more on the predictive maintenance of the hardware components, also known as condition-based maintenance. Manufacturers utilize predictive maintenance to predict equipment failures based on specific parameters and factors, which then activates the necessary steps to prevent these failures through corrective or scheduled maintenance. This research focused on analysing the consumption of different components such as motors, servos, sensors, etc., and thus the discharge characteristic of lithium batteries in autonomous mobile robots, which can serve as a model to predict future states based on the number of services running. As the robots and their services are deployed in a cloud-native extreme-edge environment where system dynamics are constantly changing, this leads to unexpected situations. In order to improve the planning and 24/7 operations of these devices, it is crucial to have a good understanding of the battery consumption and capacity degradation. To achieve this goal, three different data sets were collected, each with a different focus on the services running on the robots:

a. All services running on a robot, including image and video processing and object detection.
b. Some selected services run on a robot, with an emphasis on services with high computational cost, such as image processing.
c. Only the essential services required for movement and navigation run on a robot.

Figure 7-4 depicts a plot of the battery consumption tests conducted on various services and robots. The plot displays the relationship between battery consumption and the services running on the robots. The data points on the graph highlight the variability in battery consumption for different service combinations. The visual representation of the data allows for easy analysis and interpretation, providing a clear understanding of the impact of service selection on battery performance and illustrating the different patterns. This information can be used to inform decision-making processes around service placement and resource utilization, helping to ensure smooth and efficient operations in battery-powered systems. The goal of this research was to create a predictive approach to reallocating critical services to other robots, edge compute devices, or the cloud, to avoid service disruptions and improve overall operations.

With the training score of 0.04 RMSE and a test score of 0.21 RMSE, the predictive orchestration model for (a), shown in Figure 7-4, has a low root mean squared error on the training data and a moderate root mean squared error on the test data. The lower the root mean squared error, the closer the predicted values are to the actual values, indicating a more accurate model. Trying to further analyse the data and smooth them with average and median filtering to remove the noise from the battery levels that have been collected, the models can be trained again. The LSTM model for battery consumption predictions for (a) has a training score of 0.02 RMSE and a test score of 0.19 RMSE. This means that the model is able to accurately predict the battery consumption with a small error margin during training, but its performance is not perfect on unseen data (test data) but good enough to trigger the orchestration mechanisms when needed, e.g., before the level of the battery hits a critical point. An RMSE score of 0.19 indicates that, on average, the model's predictions are off by 0.19 units. It's important to consider this result in the context of the specific problem, where the accuracy levels are not so important and with more live data, historical data and real-time training, these results will be even better. Similar results were observed when analysing additional metrics such as motor stress, navigation error, etc.

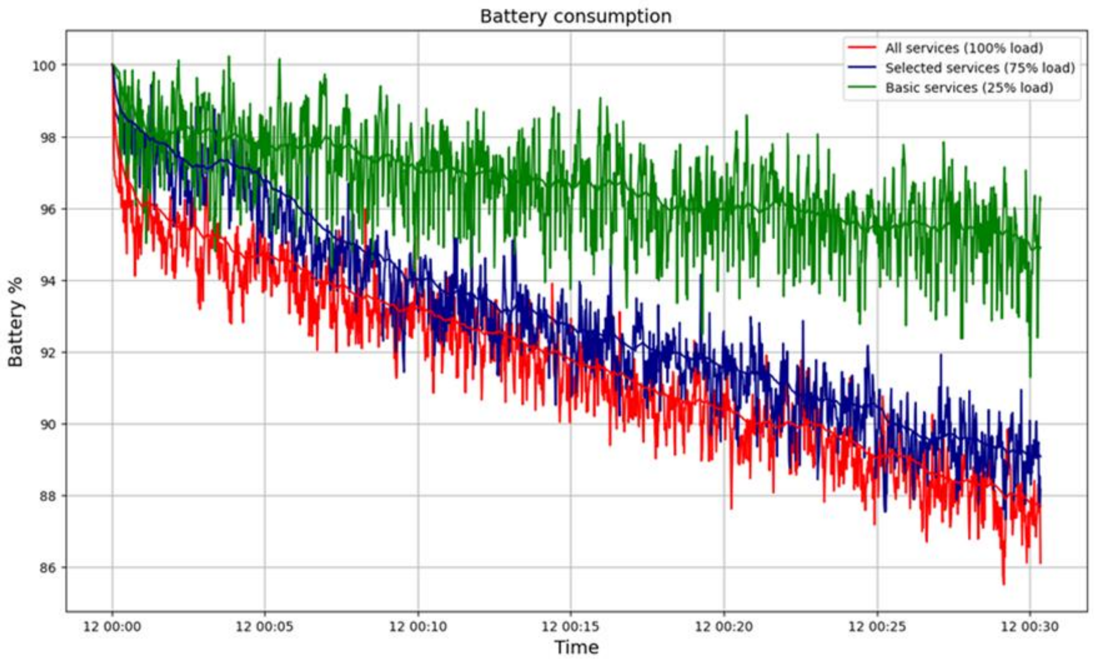


Figure 7-4. Battery consumption collected data sets.

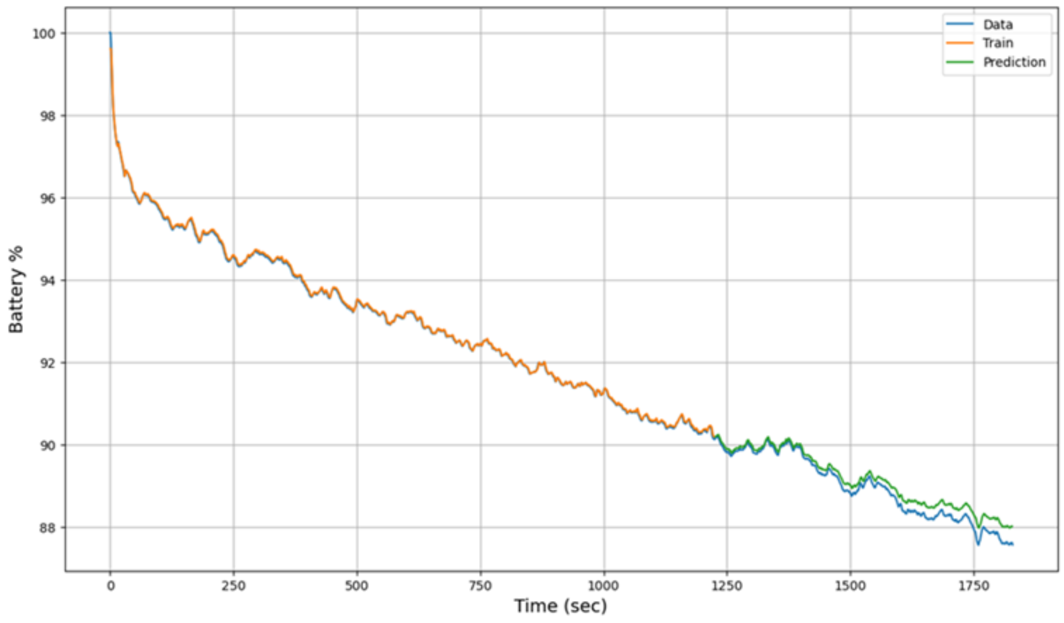


Figure 7-5. Battery consumption prediction horizon.

This predictive approach, as demonstrated in Figure 7-5, can save valuable time by proactively reconfiguring service and network components, avoiding production stops, and minimising maintenance costs compared to the reactive approach of fixing or replacing faulty components after they fail. In short, this demonstration concludes that the predictive orchestration approach is able to fulfil the QT of achieving network reconfiguration times to be performed by >10% of the prediction. An additional performance improvement caused by the introduced components and architecture is the reduction of service downtime by more than >10%.

Validation in Scenario 5.2

Regarding Scenario 5.2, the target is interpreted as being able to provide (at least) a 10% improvement in the required orchestration actions. It should be noted that any comparison, i.e., any improvement, should also take into account if the performance obtained by the application (during this orchestration) does not fall below its minimum requirements for a satisfactory operation. This is relevant since variations in the network state or load can derive in poor network performance, and therefore the intelligence of the network is critical to ensure that the URLLC requirements are always fulfilled. To analyse and put in context the resulting performance of the orchestration actions, four different methods have been implemented (the first three methods are benchmarks, while the fourth one is the proposal):

- Pure reactive: the orchestration is triggered only when servers are full.
- Oracle: complete knowledge of the future.
- Threshold-based: orchestration is triggered based on an occupation threshold.
- AI-based prediction: the proposed scheme.

Their performance is analysed as follows: it is assumed a B5G network scenario with a varying number of users, which dynamically request services to an edge system. The latter is composed of two MEC hosts (*mecHost1* and *mecHost2* in Figure 7-6) that can be activated dynamically. Each MEC host can serve the users' requests, exhibiting a service performance that depends on the total number of services currently on the host itself. Time is divided into slots with fixed durations (1s in the following examples).

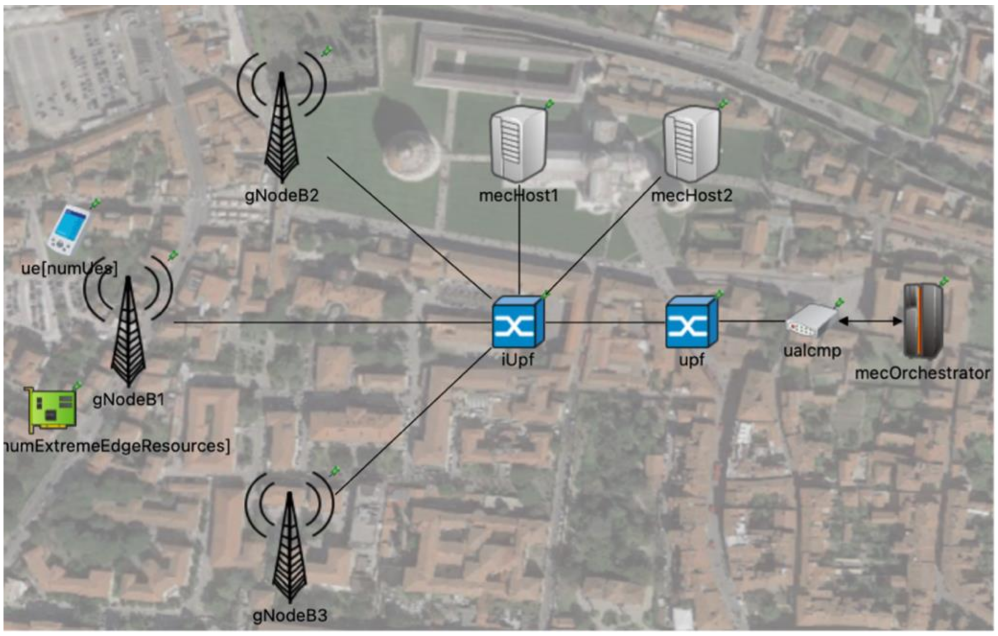


Figure 7-6. Simulated deployment for the experiment of Scenario 5.2.

The figures below depict the resulting performance of each method. For each plot,

- the red line represents the actual traffic in the system in terms of the number of active services;
- the green line represents the activation decision taken by the orchestration algorithm, with values 1 (activate a node), -1 (deactivate a node) and 0 (do nothing);
- the blue and yellow lines represent the number of services allocated respectively to the hosts *mecHost1* and *MecHost2* (if active).

The performance is compared in terms of the average *delta reconfiguration time (DRT)* of an orchestration solution, which is defined as the time distance in the number of slots between the instant of the orchestration decision and the optimal reconfiguration instant (as selected by the oracle). The results are as follows:

- The pure reactive approach has a DRT of 1 slot, although it fails to guarantee performance to the URLLC application, so it cannot be considered for comparison.
- The threshold-based solution has a DRT of 5.5 slots and is able to guarantee performance, although it performs significantly more orchestration actions.
- The proposed AI-based Prediction approach has a DRT of 0.25 slots.

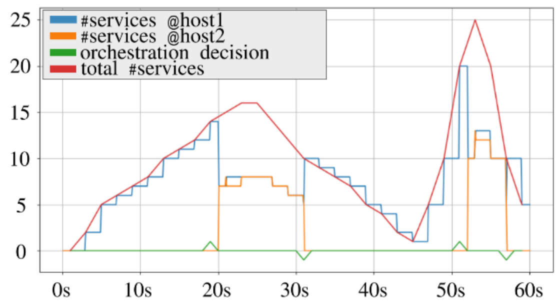


Figure 7-7. Behaviour of the *pure-reactive* baseline in Scenario 5.2.

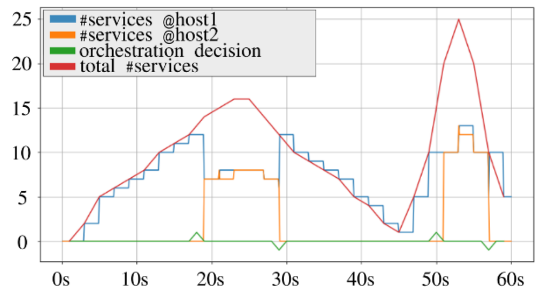


Figure 7-8. Behaviour of the *Oracle* (theoretical optimum) baseline in Scenario 5.2.

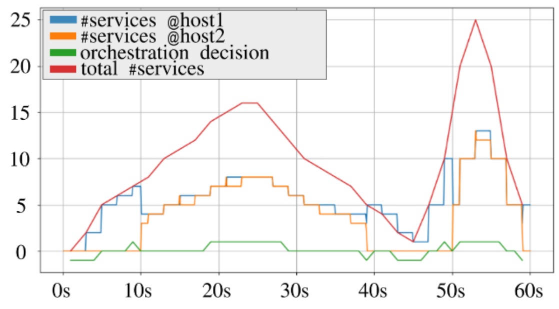


Figure 7-9. Behaviour of the *Threshold-based* baseline in Scenario 5.2.

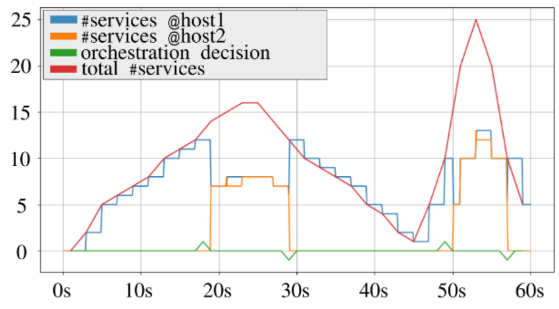


Figure 7-10. Behaviour of the *AI-based prediction* in Scenario 5.2.

Based on these results, it can be concluded that the proposed approach reduces by > 90% (0.25/5.5 slots) the delay to perform the optimal orchestration decisions, so the QT can be fulfilled.

7.1.3.2 QT 3b. Improvement in time to onboard/remove resources from other domains

This QT requests to validate the feasibility of improving by (>90%) in time the onboarding of new resources from other domains and managing the addition/removal of elements from the network. As for the other QTs, it is considered not possible to provide a general answer to this target without defining a clear baseline, so a specific test case has been used to demonstrate that the target can be achievable, at least in a specific scope. In this case, the tests performed have been in the context of what is addressed in Scenario 5.4 (MLOps), where two different domains are considered (as requested in the QT statement): the MNO and the SW Vendor domains (see Section 5.3.4), and where the cloud-native DevOps practices are applied to showcase the onboarding of an AI/ML model from the SW Vendor domain into the MNO domain, using automated workflows for that[13]. So, for this QT, the considered baseline is hence the legacy approach, i.e., the deployment of a service without using these cloud-native DevOps methodologies, which are compared with the results obtained by applying the MLOps techniques in the aforementioned Scenario 5.4[14]. For this baseline, the information provided in a previous project granted by European Commission (NGPaaS) are used, specifically in [NGP18-D31], where the MNO participating in that project provided detailed information about their general service deployment workflow (see Section 2.2 in [NGP18-D31] - Today's Networks Development Model from a Telco Provider), including their interactions with the SW Vendors. As mentioned

[13] Although the QT statement mentions onboarding and removal, the focus is on the onboarding stage here, since it is considered the most challenging one: in the DevOps approach the onboarding includes also de development phase (performed at the SW Vendor domain), while the removal is just about to eliminate the service at the MNO scope, which of course requires much less time than the onboarding of a new service.

[14] Please note that in this context the terms DevOps and MLOps are used in an analogous way. This is because, in what regards this QT, MLOps can broadly be considered just as a particularisation of DevOps, targeting the development and operation of AI/ML-enabled services. In both cases (MLOps and DevOps) one of the key features is the high degree of automation in implementing the development and operational workflows, which is what is considered the key aspect regarding the achievement of this QT.

in that reference, “the process, without applying any DevOps strategy, can take a considerable amount of time and could last up to 3 years from the initial code development until the live deployment phase”.

In the experiments conducted in Scenario 5.4, the total time required to complete the cycle and, therefore, have a model being served in the production environment depends on two main workflows, namely the development and validation workflows (see Figure 5-27 in subsection 5.3.4.1). In our lab experiments, the development workflow takes 4 hours approximately. However, it has to be clarified that this does not include the “ML Model design” since the demo assumes a pre-designed model. Of course, in a real-life scenario, this process can take a highly variable amount of time, depending on the specific problem to be solved, the development team in charge, and other factors.

Regarding the validation workflow, the approach should be similar: in the demo, this is done in quite an agile way based on automatic test batteries, taking a few minutes for it (the demo is intended to be showcased in a short span of time). However, in real-life scenarios, testing procedures can obviously last for much longer times, especially if non-automated tests are also carried out. In addition, considering the times measured in the demos, it can be seen that the duration of the whole process could be reduced to just a few hours (4 hours for de development/training plus a few minutes for the testing) without considering the offline executions (design and possible other offline testing procedures). Anyway, the result is well below those three years reported by BT.

The large time difference between both scenarios suggests that the proposed target (gain greater than 90%) may indeed be achievable, at least in certain specific cases. In fact, the same project mentioned above (NGPaaS), which also considered the application of DevOps-like techniques in the telco-grade environment, reported times also in the range of a few seconds or minutes for the KPIs regarding the service development and deployment workflows, considering services for different scenarios (5G and IoT) and several network service components (an AMF, an HSS and an IoT component) [NGP19-D32]. This also reinforces the idea that the target is, in fact, achievable. However, further research should be done on more specific ways the DevOps methodologies could be applied in the telco-grade environment since the regular DevOps approach is typically applied “within” single organizations, promoting the joint work of their operational and development teams. The telco-grade ecosystem is different in this sense since, in this case, different independent corporations and stakeholders have to work together to develop and deploy the network services to be provided by the MNO. In this regard, this way of “working together” the development and operational teams promoted from the DevOps approach can be more challenging in the telco-grade environment (as it has been approached in Scenario 5.4), which probably would require a clearer definition of how the interactions among the different stakeholders should be performed, mainly when different SW providers must be involved to develop and deploy complex services.

7.1.3.3 QT 3c: Improvement regarding service continuity

As explained in Section 5.3.3, 6G communication networks can be subject to cyber-attacks. Those attacks may directly disturb service continuity, and this is typically the case with DoS and DDoS attacks. But the service can also be willingly shut off by the owner due to other forms of attacks, such as data leakage or intrusion causing unexpected behaviour. In any case, being able to detect and remediate the attack as fast as possible minimizes service disruption and improves service continuity. To optimize the reaction time, a layered security architecture is proposed, with the lower layer deploying fast but not necessarily optimal mitigation actions with the upper layer working in parallel to deploy more complex eradication strategies. In this deliverable, the improvement in service continuity is considered from the security point of view: the objective is to react as fast as possible to a security event to limit its impact on the service. Indeed, an attack may cause the service to be unavailable for a given time, either due to the effect of the attack itself or due to the service owner willingly stopping or reducing the service to avoid potential damages caused by the attack. This is taken into consideration in Scenario 5.3. As described in Section

5.3.3.3, this fast reaction response need is a primary concern for the proposed solution: while the central loop focuses on providing a long-lasting solution for the incident through eradication, in parallel, the local loop is dedicated to providing a temporary solution as fast as possible through mitigation. In order to measure the effectiveness of this system, we measured the relevant response times of our system:

- Local Mean Time To Detect (Local-MTTD).
- Global Mean Time To Detect (Global-MTTD).
- Local Mean Time To Respond (Local-MTTR).
- Global Mean Time To Respond (Global-MTTR).
- Extended Mean Time To Respond (Extended-MTTR).

In the context of Demo #5, Local-MTTD (resp. Global-MTTD) is the elapsed time between the emission of the attack packet by the rogue UE and the production of the corresponding event by the Local (resp. Global) Monitoring and Analysis on the relevant Kafka topic, which characterizes the attack and makes the alert public for any relevant consumer.

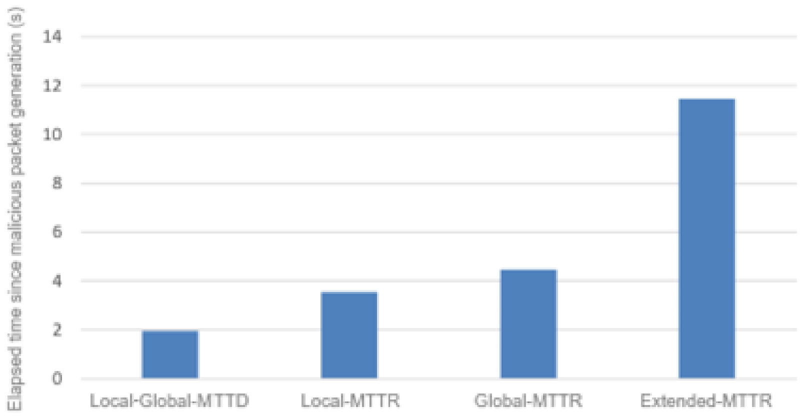


Figure 7-11. Scenario 5.3 results

As events produced in Kafka are timestamped by default, the measure relies on Kafka's timestamp. In the specific attack considered in Scenario 5.3, local- and Global-MTTDs are very close since the global analysis module does not perform additional analysis to further characterize the attack. Thanks to the efficiency of Kafka, the results show that the difference between local and global MTTDs is actually less than a millisecond; hence both will simply be referred to as MTTD in the remainder of this Section. The Local-MTTR is the elapsed time between the emission of the attack packet by the rogue UE and the activation of a rule to reject log4j packets in Suricata IPS. This activation is considered effective when Suricata's log indicates that the rule set has successfully been reloaded (after the addition of the new rule). Global-MTTR measures wait for the patch to be active. As the patch is applied by a K8s Ephemeral container, the patch activation is validated by the log output of the Ephemeral container itself. Since the timestamps of the different events used to evaluate our metrics are collected in different machines, those machines need to be strongly synchronized to have precise measures. To do so, the different machines in OpenStack are synchronized using Network Time Protocol (NTP). The reference clock is an external one, and its values are retrieved by the UPF. The results obtained are represented in Figure 7-11Figure 7-11. As we can see, the different metrics are quite close in time. This was expected, as the eradication action is fairly simple, considering that the pod to be patched is a demonstration pod. To dive more into the details of our results, we can note that:

- The time elapsed between the attack launch and the report of this attack in Kafka by the analytic engine, which constitutes the MTTD, is around 62.3s. This time is almost entirely spent in Suricata.
- It takes approximately 300ms to complete the containment once the attack report is published (Local-MTTR – MTTD).

- It takes approximately 1.3s to complete the eradication once the attack report is published (Global-MTTR – MTTD).

7.1.3.4 QT 3d. Improvement of the network energy efficiency using predictive orchestration

This QT requests to validate the feasibility of *increasing the network energy efficiency by (>50%) applying predictive orchestration* techniques. As for the previous QTs, and as explained in 7.1.3, it is considered not feasible to validate that statement in a general way without specifying a clear benchmark. For this specific case, the quantitative result may depend on a multiplicity of factors, such as the technologies in use (e.g., the specific M&O software platforms), the deployed network services (energy usage can obviously vary from one service to another), the hardware resources on which the software components are deployed, or the specific predictive algorithms in use. I.e., it is obviously not possible to claim that applying "predictive orchestration" (in general) will "always" entail an improvement of the network energy efficiency in the requested percentage (>50%) without considering what specific services, specific M&O systems and mechanisms, hardware, or specific predictive algorithms are being compared. So, in order to address this QT, it was considered that anyway it could be demonstrated that the requested improvement in energy efficiency could be achieved considering a set of scenarios that, although it cannot be generalised, can at least give an idea about whether the target can be achieved. In order to do this, a specific lab experiment (the one described in the previous Section 6.1) has been performed.

Based on that experiment, it can be stated that by adapting to the forecasted demand, it is actually possible to reduce the power consumption to approx. 324.769 W, which represents approximately 88% of the original energy consumption, beyond the QT definition requires. However, as mentioned, this is just a specific lab experiment, difficult to generalize. Probably this 12% reduction can be harder to reach in other scenarios with multiple factors. Further research should be done once the 6G network stack is available, as well as the specific 6G M&O platforms.

7.2 Validation of the Hexa-X M&O architecture

This section focuses on evaluating whether the M&O architectural design provided in Deliverable D6.2 [HEX22-D62] has proved suitable for the demos' implementation. Although the complete E2E architecture will not be validated (due to the specific scope of each demo), it is considered that the results obtained here can be a good indication of the possible application of this architectural design in future 6G networks.

7.2.1 Demo #4

It has been verified that the various functionalities implemented in this demo can be mapped with the M&O architectural design of D6.2, as illustrated in Figure 7‑1. The vertical application, Digital Twin App, is part of the Service Layer (1) as the main Verticals-focused layer. The components that are part of the intelligent orchestration group belong to both the Service Layer and the Network Layer (2, 3, 4, 5). These components are tasked with the orchestration, diagnostic and monitoring aspects of the demo. Finally, the Monitoring framework, along with the monitoring agents (probes), belong to both the Network and the Infrastructure Layers (4, 6, 7).

For the purposes of this demo, different software components have been developed for the Service, Network and Infrastructure layers, as described below.

- Service layer. In this layer, the Digital Twin application, developed for this demo, is located. This service is common across the three scenarios and is used to support the vertical's industrial automation service.? The implemented service layer control loops are focused on validating the service requirements and SLAs and triggering quality assurance mechanisms when these are not honoured.
- Network layer. It is partially implemented, meaning that there are no components deployed as part of Radio Access, Core Network, Transport Network and Third-Party Functions as they were out of the scope for the purposes of this demo. The orchestrator has been extended with

higher-level logic in order to allow the management of extreme-edge resources as part of the *Management Functions*. Part of these functions is also the *Functions Placement* mechanism that is responsible for deciding the optimal way to place the various functionalities based on predefined policies like power consumption, nodes in use, etc. The Service Registry and Service Repository are also part of the Management Functions block in this layer. They are responsible for the registration, storage and retrieval of the necessary information pertaining to the various services (requirements, descriptors, etc.). Additionally, the *Monitoring Functions* that have been implemented include the use of the *MaaS framework* that allows the advanced monitoring of various cloud, edge, and extreme-edge resources. This framework consists of a central monitoring solution that handles the deployment of probes on the distributed resources across the different managed domains. Finally, the *AI/ML Functions* that have been implemented include the *Diagnostic* component and the *Predictive Orchestration* mechanism. The Diagnostic component is responsible for detecting anomalies in the observed performance of the system under examination, service or infrastructure, detecting the root cause of that anomaly with sufficient information and triggering corrective actions. On the other hand, *Predictive Orchestration* uses historical data to predict the future state of the system and trigger proactive orchestration actions in order to minimize service downtime.

- <u>Infrastructure layer.</u> This layer is implemented using the physical and virtual hosts described in Table 4-1 for the Cloud and Edge domains as well as the robots for the extreme-edge domain.

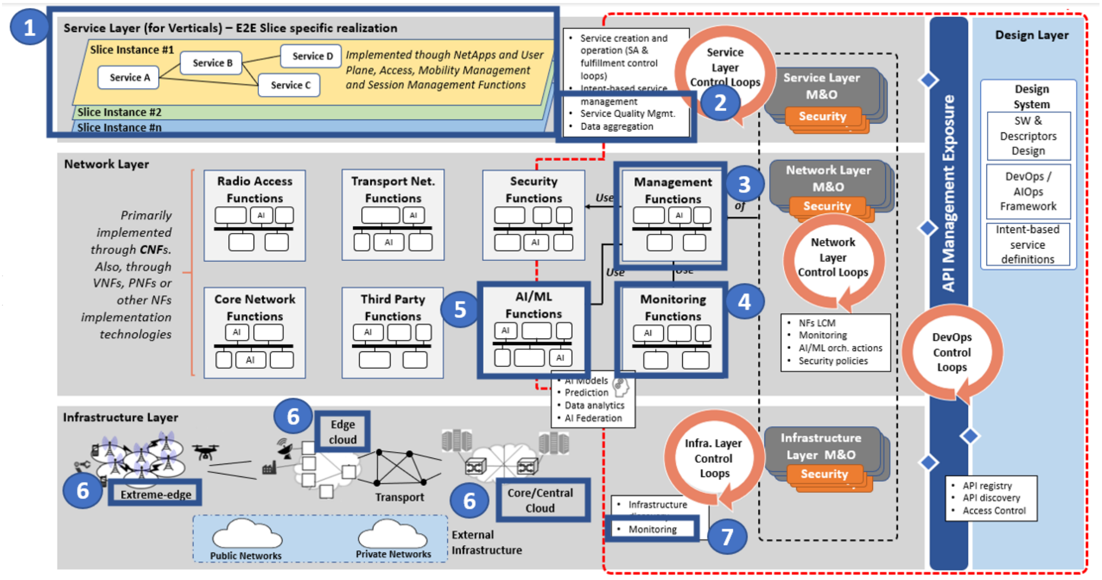


Figure 7-12. Mapping of the Demo #4 functionalities to the architecture.

7.2.2 Demo #5

As described in Section 4, Demo #5 aims at demonstrating a set of pertinent features, targeting the data-driven device-edge-cloud continuum M&O concept by means of four specific demo scenarios. Figure 7-12. maps the different functionalities implemented across these four scenarios into the Hexa-X M&O architectural design provided in [HEX22-D62], so as it can be seen, this architectural design can actually be used as a framework to support the different functionalities targeted in the demo. In the following, a detailed per-scenario description of the mapping is provided.

<u>Scenario 5.1</u>

This scenario incorporates resources across every Infrastructure Layer domain (circles '1a', '1b' and '1c' in Figure 7-13.) because it needs to deal with elements that are allocated within the Extreme-edge domain (e.g., K3s), the Edge domain (e.g., AI agents and the message queue) and the Cloud domain (e.g., the service orchestrator). Moreover, at the Network Layer, the orchestration platform provides capabilities to cope with the scenario NFs LCM (5) and Monitoring (6). At the Design Layer level (7), the orchestration platform is able to provide Service definition blueprints based on K8s deployments that aid in the service creation and its LCM (10). Finally, a Slice (8) is the communication baseline for all the services deployed within this scenario across all the aforementioned Infrastructure Layer domains. The orchestration platform deployed within this environment (the Vertical Slicer) implements the required M&O functionalities for this scenario, which are mainly focused on service deployment.

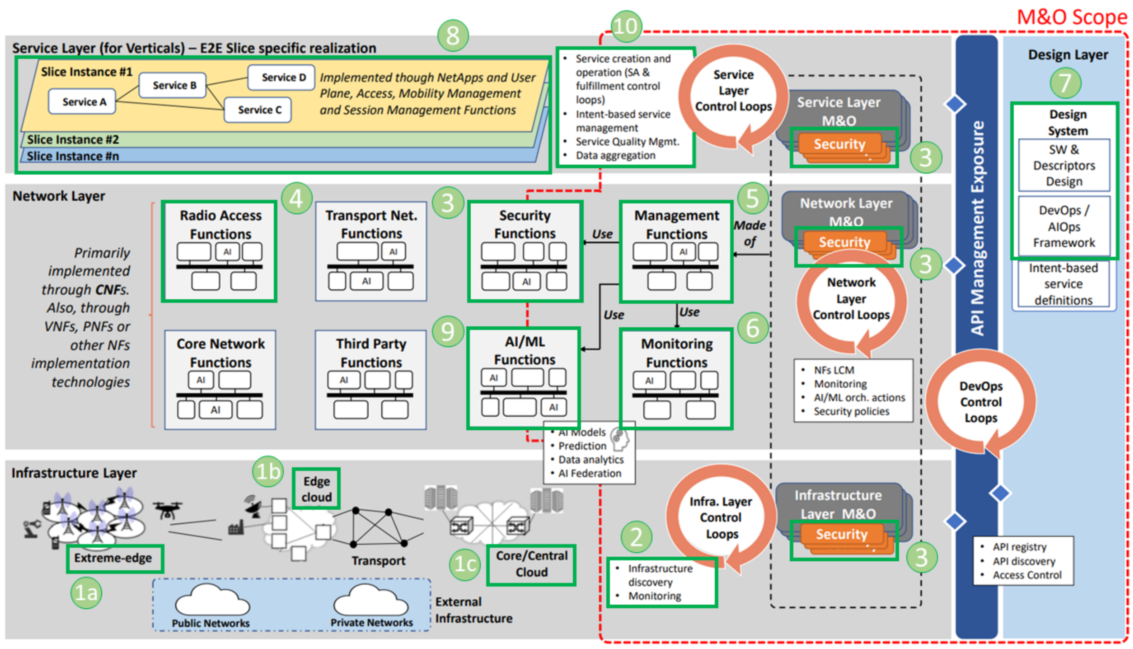


Figure 7-13. Mapping of Demo #5 functionalities to the Hexa-X M&O architecture

Scenario 5.2

This scenario has similarly mapped elements to the ones described in the previous scenario as it tries to complement it by adding *Predictive Orchestration* capabilities. As depicted in Figure 7-13., all Infrastructure Layer Domains are emulated in this scenario (1a, 1b, 1c), but only the monitoring capability from the Infrastructure Layer (2) has been added to the M&O function in this scenario. The Management (5) and Monitoring Functions (6) building blocks have also been deployed as part of the so-called *Orchestration Engine* at the Network Layer. Furthermore, on a regular basis, the so-called *Intelligent Orchestrator component*, which maps to AI/ML Functions *(9)*, receives input from the Orchestrator Engine component, and within the simulated network, several AI-based orchestration actions are taken. At the Service Layer, the modelled *URLLC Application (8)* is deployed across UE devices and the edge server (see Section 5.3.2.2). To end, Simu5G allows the simulation/emulation of UPFs, PGWs and gNBs, which interact using a model of the NR protocol stack. These components are used within this scenario as part of the Radio Access Functions (4).

Scenario 5.3

Scenario 5.3 is focused on validating the Hexa-X M&O architecture from a security perspective at each layer. Specifically, it addresses the implementation of the LoTAF [HEX22-D14] at the Service Layer (3) and the deployment of local (1a) and central security orchestration functions (1b, 1c) at the Infrastructure Layer. When it comes to the application of AI for security (9), the Central Security Orchestrator also centralises AI/ML training tasks. The information produced by the Local and Central Control Loops might be consumed across every layer in the M&O architecture (3).

Scenario 5.4

Scenario 5.4 studies the feasibility of applying MLOps for a particular use case where an SW Vendor develops, trains and deploys an AI/ML-based model on the MNO infrastructure. The resources involving the implementation of the MLOps platform have been deployed both at the edge domain (1b) and the cloud domain (1c). The main focus in this scenario is, in fact, to demonstrate the integration of the Design Layer (7) as part of the M&O architectural design, with the deployment of an AI/ML model using MLOps techniques, i.e., the demo is designed precisely to demonstrate one of the innovative features of the architecture proposed in D6.2. As can be seen, the Hexa-X M&O architecture can be used as a reference M&O architecture for B5G/6G use cases, and it is flexible enough to cope with the different requirements and constraints of each particular use case. Specifically, in the scenarios comprising Demo#5, several building blocks comprising this architecture have been validated, and therefore, the utility of this architecture for future 6G mobile networks has been probed.

7.3 KPIs, KVIs and Core Capabilities

Below are the main KPIs (from those in the previous D6.2 [HEX22-D62]) measured on each demo and lab experiment. The KPIs are also related to the relevant KVIs and Core Capabilities defined globally in the Hexa-X project.

7.3.1 Demo #4

The three scenarios of this demo, while having different objectives, all share the same target KPIs, KVIs and Core Capabilities. Below is the information for each of these three scenarios' objectives in this demo. Afterwards, the KPIs, KVIs and Core Capabilities are described in a concise way.

Scenario 4.1

The main objective of this scenario is to analyse the impact of Scenario 4.1 software components on a B5G/6G platform. The analysis revolved around the extended programmability provided by a unified Cloud – Extreme-edge continuum. The focus of this evaluation is on the following:

a) the level of automation that can be achieved utilizing the extended programmability offered by the various implemented software components;
b) the inclusion of extreme-edge nodes in the orchestration workflows and their performance during the industrial context workflows.

Scenario 4.2

The main objective of this scenario is to analyse the impact of Scenario 4.2 software components on a B5G/6G platform. The impact has been studied for its feasibility of utilizing advanced monitoring and diagnostic as well as Function Placement mechanisms in order to optimize performance related to the workflow time pertaining to service orchestration actions. These actions are based on the observed performance and aim to minimize the downtime of said services due to M&O operations. The focus of this evaluation is on the following points:

a) similar to Scenario 4.1, the level of automation that can be achieved utilizing the extended programmability offered by the various implemented software components;
b) the performance of the extreme-edge nodes, and robots, when running industrial automation tasks;
c) the performance of the diagnostic mechanisms to detect anomalies and optimize Function Placement.

Scenario 4.3

The main objective of this scenario is to analyse the impact of Scenario 4.3 software components on a B5G/6G platform. The impact has been studied for its feasibility of utilizing predictive mechanisms in order to optimize performance related to the workflow time pertaining to service orchestration actions. These actions are based on the predicted performance of the services and aim to minimize the downtime of said services due to M&O operations. The focus of this evaluation is on the following main points:

a) as in both of the previous scenarios, the level of automation that can be achieved by utilizing the extended programmability offered by the various implemented software components;
b) the performance of the predictive mechanism when predicting the future state of the service, performance degradations and upcoming maintenance cycles.

The KPIs, KVIs and Core Capabilities for the three scenarios are presented below.

KPIs
Programmability [%]: In Scenario 4.1, the programmability of the compute infrastructure in the extreme-edge, edge and cloud domains is utilized in coordination with the monitoring platform of the virtual resources available in each node and for the management of the service components in the various clusters. Moreover, the interfaces exposed by the orchestration platform handling the compute resources located on top of the industrial robots enable the registration of nodes belonging to the related cluster and the continuous monitoring of their status, reachability and availability. In Scenario 4.2, the utilization of the infrastructure monitoring and diagnostics mechanisms is exploited in order to drive the optimization of the Function Placement across the domains. Moreover, the M&O operations are examined in order to evaluate the performance of the newly introduced intelligent orchestration processes. In Scenario 4.3, the programmability of the cloud - extreme-edge continuum, and especially the utilization of the predictive orchestration mechanism, is exploited to optimise the Function Placement across the domains. Moreover, the performance of M&O operations is examined in order to evaluate the efficiency of the predictive mechanism. All the M&O mechanisms exploit the provided programmability of infrastructure and service components, and as such, the level of programmability can be considered at 100%.
Processing Capacity [Number & Type of processing units]: For all three scenarios, the industrial automation service is deployed across extreme-edge, edge and cloud nodes, involving in particular: • Three robots (equipped with quad-core Intel processors @ 3.6GHz and 8GB RAM), running Ubuntu v20.04, and organized in an extreme-edge K3s cluster. • Two cloud/edge nodes based on two general-purpose HP computers (Z2 Mini G4) used for VM-based and container-based components, respectively.
Creation time [s]: In Scenarios 4.1 and 4.2, the lifecycle management of the industrial automation service is automated through the orchestrator and does not require human intervention, thus reducing the orchestration actions workflow time. The instantiation time of the service ranges between 3 and 60 seconds. Typically, it takes 60 seconds when starting the service in a new context, meaning a new node that it has not been used on before and does not have the images available locally. After the service has started on that node at least once and the images are available, the instantiation time falls to a stable 3 seconds. In either case, additional 2 seconds are required for the service configuration bringing the total to around 5 seconds. Additionally, in Scenario 4.3, since in this scenario, a proactive orchestration approach is used, the creation time does not impact the service, in performance or downtime, because all the required actions are taken while the status of the examined services follows their requirements.
Automation [degree]: All the steps of the demonstration are fully automated and coordinated through the orchestrator, with the only exception of the service instantiation request, which is triggered manually via a REST API. For this use case, the available extreme-edge nodes are considered already registered on the platform and so are taken into account by the orchestrator as available resources. At the service level, the industrial automation service is a predefined set of functionalities, separated between controller tasks and worker tasks, that target different domains to be deployed on. The selection of the appropriate domain and target infrastructure for the deployment is handled by the intelligent orchestration set of mechanisms that is demonstrated in the next scenarios. For Scenario 4.1, the functionalities are allocated uniformly across all the available resources. The configuration of the service is handled internally by the service itself, which adjusts its operation based on the available resources. In scenario 4.2, manual injection of impairments are utilized to emulate the real impairments that would have happened without manual intervention. These impairments trigger the diagnostic

and Function Placement mechanisms that further demonstrate the high degree of an automated process that is responsible for rectifying the anomalous condition and restoring the service to its correct state.

In scenario 4.3, the AI models used to predict the future state of services have been trained during the normal industrial task cycles and are utilized in order to evaluate the prediction mechanism. The inference, decision for actions and feedback operations of the predictive mechanism all operate automatically without human intervention. All the aforementioned mechanisms and processes bring the automation degree to a "high".

KVIs

Trustworthiness: It is based on the high level of automation in the management of the resources in the continuum and in the provisioning and lifecycle management of the end-to-end service across all three scenarios.

Sustainability: This KVI is targeted by three features:

- First, in Scenario 4.1, the automation of the M&O of the resources removes the requirement for human intervention while providing the capability for it if need be.
- Second, in scenario 4.2, the utilization of the Function Placement mechanism optimizes the resource usage across the available domains while targeting specific policies like minimizing power consumption, resource usage or any other provided requirement.
- Third, in Scenario 4.3, the utilization of the predictive orchestration mechanism optimizes the M&O actions required to prevent "predicted" anomalous events based on the trained observed performance patterns as well as operation and maintenance cycles.

Core Capabilities

Integrated intelligence: Scenario 4.1 demonstrates the logic integrated into the orchestrator, which handles the service lifecycle management, including the coordination of the instantiation and configuration steps and the management of the resources in the continuum, with extreme-edge resource registration and monitoring of the available resources. Scenario 4.2 demonstrates the logic integrated into the orchestrator, which is responsible for handling anomalies in the observed performance and also optimizing the Function Placement of services and their components based on desired policies. Scenario 4.3 demonstrates the logic integrated into the orchestrator, which is responsible for predicting the future state of the service performance along with the operation and maintenance cycles and also optimizing the Function Placement of services and their components based on these predictions.

Usage of Embedded devices: Scenario 4.1 has been selected to showcase the M&O of resources running on industrial robots as extreme-edge nodes organized in a K3s cluster. Scenario 4.2 demonstrates the logic integrated into the orchestrator, which is responsible for handling anomalies in the observed performance and also optimizing the Function Placement of services and their components based on desired policies. Finally, Scenario 4.3 has been specifically designed to demonstrate the possibility of optimizing the operation and maintenance cycles of the extreme-edge devices (robots) based on the predicted performance state during said cycles leading to reduced costs, minimized service downtime and optimal service performance.

Flexibility: The orchestrator demonstrated in Scenario 4.1 is flexible enough to operate over a variety of extreme-edge, edge and cloud domains, deploying K3s, K8s and OpenStack platforms. Moreover, the orchestrator can successfully instantiate and configure end-to-end services based on a mix of container-based and VM-based components. As demonstrated in

Scenario 4.2, the orchestrator can successfully optimize the placement of industrial automated tasks on the available resources based on the service requirements. Finally, in Scenario 4.3, the orchestrator can successfully predict the future performance state of the examined services, along with the operation and maintenance cycles of the infrastructure nodes and utilize these predictions to perform the necessary M&O actions to ensure optimal performance.

7.3.2 Demo #5

Below is the information for each of the three scenarios of Demo #5.

Scenario 5.1

As described in Section 5.3.1, the main objective of this demonstration scenario is to validate the feasibility of dynamically deploying a distributed virtual application, implementing AI/ML techniques in a 6G-enabled infrastructure continuum that integrates extreme-edge, edge and cloud nodes organized in different clusters. The focus of the evaluation is on three main features: (i) the level of automation that can be achieved by exploiting the programmability of the computing nodes when creating and orchestrating a new service; (ii) the performance of extreme-edge nodes when running virtual application components; and (iii) the performance of the AI/ML functions when deployed in a distributed edge/cloud environment. Below, the KPIs, KVIs and Core Capabilities that are considered relevant in this scenario are described.

KPIs
Programmability [%]: In this scenario, the programmability of the computing infrastructure in the extreme-edge, edge and cloud domains is exploited for the monitoring of the virtual resources available in each node and for the dynamic instantiation of the application components in the various clusters. Moreover, the open interfaces exposed by the K3s platform handling the small-scale Raspberry Pi computers enable the dynamic discovery of nodes belonging to the related cluster and the continuous monitoring of their status, reachability and availability. In this scenario, all the M&O procedures directly exploit the programmability of infrastructure and service components; as such, the level of programmability can be considered at 100%.
Processing Capacity [Number & Type of processing units]: The service is deployed across extreme-edge, edge and cloud nodes, involving in particular: • Four Raspberry Pi cards (brand Broadcom, model BCM2711, with a 64-bit quad-core Cortex-A72 -ARM v8- @ 1.5GHz, and with 8GB RAM), running Ubuntu v20.04.5 and organized in an extreme-edge K3s cluster. • Two cloud/edge nodes based on a general-purpose PowerEdge T550 Dell server and an Intel NUC small-form computer (NUC8i7HVK), used for VM-based and container-based components, respectively.
AI/ML models training time [s]: Something relevant to be considered here is that, due to the application of the Reinforcement Learning paradigm, there is not a clear split between the learning and the inference stages, i.e., RL models learn at the same time they are interacting with the environment on which they are integrated (the urban road traffic scenario in this case), so they can continue learning indefinitely. However, from a practical point of view, the RL agents are considered to be "acceptably trained" once the road traffic flow is perceived much better than when the traffic control is performed based on the legacy approach, i.e., based on fixed time patterns to control the traffic lights. In practice, it has been seen this happens after about 34.000 simulation steps. On the other hand, though the simulation step duration can be interactively modified during the simulation (to increase or decrease the simulation speed as desired), it has been seen that setting about 150 milliseconds as the simulation step, the vehicles move with speed similar to how they would move in real life. So, considering these 150 milliseconds and the 34.000 simulation steps needed for the training, that gives 85 minutes as

the needed model training time. Of course, this number should not be considered narrowly since, as explained, the calculated value is based on a subjective assessment of the simulation result. However, it can give an idea of what the value of this KPI might be in a real-life implementation.
Creation time [s]: The entire provisioning of the traffic-light control service is entirely automated through the orchestrator, removing the need for manual intervention and thus reducing the creation time to less than 1 minute. In detail, the average creation time is in the order of 20 seconds, reaching a maximum of 45 seconds. Within this time interval, only 5 seconds are required for the instantiation of the containers, while the rest is for starting and configuring the application software.
Automation [Degree]: All the steps of the demonstration are fully automated and coordinated through the orchestrator, with the only exception of the service instantiation request, which is triggered manually via REST API. In particular, at the infrastructure layer, the nodes in the three domains are automatically discovered, and their availability is continuously monitored. At the service and network layer, the service request issued by the user is automatically elaborated on the basis of the service blueprint and translated into a set of components which are deployed by the orchestrator on the target domains across the extreme-edge, edge, cloud continuum. The selection of the target clusters and nodes is automated on the basis of the availability of the resources. The configuration of the application is also automated according to the service blueprint and the parameters declared by the user in the instantiation request.

KVIs
Trustworthiness: It is based on the high level of automation in the management of the resources in the continuum and in the provisioning and lifecycle management of the end-to-end service.
Sustainability: Sustainability is increased in two aspects: the automation of the resource and service management limits the need for manual intervention, while the adoption of resource allocation algorithms optimizes the usage of the resources in the various clusters and domains.

Core Capabilities
Integrated intelligence: the scenario demonstrates the logic integrated into the orchestrator, which handles the service lifecycle management, including the coordination of the instantiation and configuration steps and the management of the resources in the continuum, with resource allocation algorithms and automated discovery and monitoring of the available nodes.
Usage of Embedded Devices: the scenario has been specifically designed to demonstrate the possibility of deploying and successfully running part of the end-to-end service in four Raspberry Pi cards (with Ubuntu v20.04.5) organized in a K3s cluster.
Flexibility: the orchestrator demonstrated in this scenario is flexible enough to operate over a variety of extreme-edge, edge and cloud domains, deploying K3s, K8s and the OpenStack platform. Moreover, the orchestrator can successfully instantiate and configure end-to-end services based on a mix of container-based and VM-based components, with the possibility to define specific characteristics, constraints and configuration parameters in the service blueprint.

Scenario 5.2

As described in Section 5.3.2, the main objective of this scenario is to demonstrate the feasibility of a predictive-orchestration approach, which manages resources in the cloud-to-extreme-edge continuum. The scenario has been implemented using a hybrid simulation/emulation approach, allowing for the comparison of several baselines against the proposed solution. Below, the relevant KPIs, KVIs and Core Capabilities are described.

KPIs
Programmability [%]: In this scenario, the programmability of the computing infrastructure in the extreme-edge, edge and cloud domains is exploited for the dynamic activation/deactivation of (simulated) edge servers and for the relocation of edge services. All these operations are performed in an automated manner within the emulated environment. As such, the level of programmability can be considered at 100%.
Processing Capacity [Number & Type of processing units]: The service is deployed across extreme-edge, edge and cloud nodes, involving in particular: • A Qotom MiniPC, with Ubuntu 20.04, CPU Intel i7, 8 GB RAM, 128 GB hard disk. • A Qotom MiniPC, with Ubuntu 20.04, CPU Intel Celeron, 8 GB RAM, 58 GB hard disk. • One Raspberry Pi card (brand Broadcom, model BCM2711, with a 64-bit quad-core Cortex-A72 -ARM v8- @ 1.5GHz, and with 8GB RAM), running Raspian OS. • One server node, running the orchestration environment, with Ubuntu 20.04, CPU Intel i7, 16 GB RAM, 1 TB hard disk. • One Laptop MacBook Pro, with macOS Big Sur, CPU Intel i5, 8GB RAM, 250 GB hard disk.
Creation time [s]: Resource creation time is improved through prediction. The activation/scaling of resources is indeed made proactive, allowing the system to reduce the effects of activation times.
Automation [Degree]: All the steps in the demonstration are fully automated through the simulator's environment and are also coordinated through the orchestrator. The only exception is the user application, which has to be started manually on one of the nodes.

KVIs
Trustworthiness: It is based on the high level of automation in the management of the resources in the continuum.
Sustainability: Sustainability is increased through the adoption of predictive resource orchestration, which optimizes the usage of the resources avoiding the allocation of unnecessary ones.

Core Capabilities
Integrated intelligence: The scenario demonstrates the AI-based prediction integrated into the orchestrator, which is able to predict the onset of a load increase and thus proactively activates resources.
Usage of Embedded Devices: The scenario allows the orchestrator to offload the service to a Raspberry Pi device.
Flexibility: The orchestrator can execute several alternative algorithms, including the proposed predictive orchestration. The AI-logic, in turn, can be executed independently from the orchestration logic, allowing re-training of the AI itself.

<u>**Scenario 5.3**</u>

As described in Section 5.3.3, the main objective of this scenario is to demonstrate the possibility of applying automated security management at different layers of the infrastructure to reach an adequate security level for all services. This includes services running on extreme-edge premises, which are geographically isolated and that have few resources spared for security processes. Below, the relevant KPIs, KVIs and Core Capabilities considered for this scenario are presented.

KPIs
Programmability [%]: In this scenario, the security processes are fully programmable. In this specific case, the core of the programmability is primarily supported by the Decision Engine, Drools, which allows the administrator to define what action plans should be followed after the detection of a specific attack. The Execution engine, in parallel, relieves the administrator from the difficulty of actually implementing the details of the execution of the plan. Since programmability is at the core of the proposed system, it can be considered to be 100%.
Processing Capacity [Number & Type of processing units]: For the needs of this scenario, 6 VMs were used. The details of the resources are displayed in **Table 5-4** but sum up to 30 vCPU and 68 GB of RAM, including six vCPUs and 12 GB of RAM to deploy the data plane, which is not part of the security solution itself. It should be noted that this scenario relies partly on open-source components which have minimal resource requirement that allow them to handle much more traffic than what is needed in the scenario, but which would be useful in a real-life situation.
Creation time [s]: Considering only the deployment of the module related to the security system and not the ones related to regular user plane functions, the deployment of the whole system is automated using the helm tool. Except drools and Kafka, most of the components can be deployed in parallel, and take less than 10s to be running. Drools must wait for Kafka to run and takes 40s to be running. Kafka itself takes 80s to be fully operational in our system. Consequently, the whole security system is up and running within 2 minutes.
Automation [Degree]: The whole system is built around the concept of automated closed loops: traffic is analysed, attacks are identified, and responses are applied in an automated way. In parallel, the LoT is automatically adjusted, following the evolution of the automated security processes. However, given that security is a highly sensitive domain, automation can willingly be reduced to include human intervention.
MTTR (Mean Time To Respond): as detailed in Section 7.1.3.3, the system displays low MTTR compared to traditional human intervention. This is a major KVI as it is one of the key aspects to determine how good the security system ensures service continuity. Here the system reacts within a minute. Moreover, while most of the time is spent in analysis (MTTD), the system further improves its reaction time through its two levels reaction: while the first reaction, the containment, is applied ~1.5s after attack detection, the slower reaction (eradication) is applied after ~2.5s. Whereas both reactions display very low response time compared to non-automated security responses, the demonstration displays the feasibility and interest in decoupling different types of reactions when dealing with an attack to further gain time. Finally the proactive protection against future attacks (extended eradication) is applied ~9.5s after detection.

KVIs
Trustworthiness: In this scenario, trustworthiness can be evaluated at two levels. On the first level, the autonomic security system is designed to maintain an adequate security level, matching at least the security requirements established by the client. To do so, the system has been designed to react to attacks both in a fast and efficient way by making use of containment and eradication actions, respectively. On the second level, the actual LoT of the system is

constantly monitored. If the security system cannot maintain the required security level, this reflects in the LoT, and the service orchestrator can take action, e.g., by requesting other available function and service networks with higher LoT, which allow serving its clients.
Sustainability: Sustainability is increased through the security improvement brought by the solution. By limiting the damages caused by cyber-attacks, the whole system becomes more sustainable.

Core Capabilities
Integrated intelligence: In this scenario, the intelligence resides primarily in the decision engine, which has to select a proper action to take upon threat identification. However, this intelligence can logically be extended to the analysis engine in future developments.
Usage of Embedded Devices: While this scenario does not directly deploy services on embedded devices, it is designed to address such use cases, as it focuses on decoupling fast, simple security actions and long, resource-consuming ones. The second can naturally be hosted on a central data centre, while the first one will remain either in or as close as possible to devices hosting the services.
Flexibility: The security system runs primarily on legacy containers and is deployed via automated tools. By changing the configuration of those tools, the system can easily be deployed in different environments.

Scenario 5.4

Below, the KPIs, KVIs and Core Capabilities, which are considered relevant for this scenario, are presented.

KPIs
Programmability [%]: All the AI/ML functions have been developed with REST APIs that are used to handle their dynamic configuration and activation under the coordination of the orchestrator.
Creation time [s]: The average provisioning time for the AI/ML functions is in the order of 20 seconds, including their configuration and activation.
Reliability [%]: The reliability of the service is guaranteed at two levels. The availability of the AI/ML functions is continuously monitored by the orchestrator, that, in case of failures, can react to deploy new functions and update the overall service configuration accordingly. Moreover, the AI/ML models are continuously evaluated (in this particular example, comparing the predicted with the actual values), and, in case of drift, they are automatically re-trained on the latest version of the datasets.
AI/ML models training time [s]: Depending on the resources assigned to the VM where the target model is trained, the training time can require up to 4 hr (for 1000 epochs).
Maintainability [Degree – e.g., high, medium, low]: The high level of maintainability is guaranteed by the orchestrator's capabilities to easily restore the service as a whole or the single AI/ML functions. Moreover, the automated MLOps procedures allow immediate re-configuring of the most suitable version of the trained models in case of any disruption of the software instances.
Automation [Degree]: The entire MLOps process is automated, starting from the initial training of the model in the software developer environment and up to the model's transfer in the MNO's staging and production environment, where it is continuously validated against the

real-time data. In case of drift, also the re-training procedure and the update of the model's version for the service runtime are handled in an automated manner under the coordination of the orchestrator.

KVIs
Trustworthiness: It is enhanced by relying on the high level of reliability and maintainability of the service, as well as its degree of automation.
Sustainability: The MLOps chain is fully automated, thus requiring a minimum level of manual intervention.

Core Capabilities
Integrated intelligence: The scenario demonstrates the MLOps orchestrator logic, which integrates the coordination between the deployment of the AI/ML functions in the target environments (staging and production) and the configuration/activation of the various modules and trained models. Moreover, based on the feedback received from the continuous evaluation of the models, the orchestrator is also able to trigger re-training actions whenever a drift is detected.
Flexibility: The MLOps procedures demonstrated in this scenario are flexible enough to be adopted for various ML models and functions, with the only constraint of the availability of the required datasets and the programmability of the custom AI/ML functions.

7.3.3 Small-scale lab experiments

Experiment 1: Network Energy Efficiency.

This experiment focuses on providing the required levels of reliability and latency for a particular V2X scenario. It also considers minimizing energy consumption while ensuring the requirements. For such a case, a recurrent neural network is implemented and compared again to a purely reactive and Oracle-based approach. Below, the set of KPIs, KVIs and Core Capabilities relevant to this experiment is presented.

KPIs
Latency [s]: In this experiment, we measure end-to-end latency between vehicles and servers to ensure that it is below certain levels. For example, for the teleoperated driving service considered in the experiment, the latency must always be below 100ms. This can be extended for other vehicular services, such as cooperative driving or hazard warning, which require other levels of latency.
Reliability [%]: In this experiment, we measure reliability as the percentage of packets that are below a certain level of end-to-end latency. For the teleoperated driving service considered in this experiment, it is necessary to ensure a reliability of 99.999%. This means that 99.999% of the packets satisfy the end-to-end latency of 100ms.
Energy efficiency [W]: In this experiment, we focus on energy efficiency as one of the most important KPIs to evaluate. We define energy efficiency as the power consumed by the deployment of resources. Thus, efficient deployment of resources results in a more energy-aware approach. To evaluate this, we compare an AI/ML approach with a purely reactive and oracle-based approach.

KVIs

Trustworthiness: It is improved by the high levels of reliability ensured by the vehicular service. It is also enhanced by the high degree of automation in deploying resources at the edge.
Sustainability: The use of AI/ML techniques to deploy resources at the edge result in a more sustainable network due to the high levels of automation.
Integrated intelligence: The experiment demonstrates that AI/ML-based orchestration mechanisms can be integrated into the network to predict the upcoming traffic load and proactively scale the corresponding number of resources.
Flexibility: The automatic deployment of resources by AI/ML orchestration algorithms enhances the flexibility of the network. The orchestrator can also execute many different algorithms.

Experiment 2: Extreme-edge nodes discovery

This experiment focuses on a particular aspect of the demonstration scenario 5.1, i.e., the dynamic discovery of extreme-edge nodes. As such, it mainly addresses the aspects related to the programmability of the platform and its level of automation for resource orchestration. It should be noted that since this experiment has been performed with emulated nodes, the considerations about the processing capacity of extreme-edge nodes are not applicable. Below are the KPIs, KVIs and Core Capabilities that are considered relevant in the scope of this experiment.

KPIs
Programmability [%]: In this scenario, the programmability of K3s-based computing infrastructure in the extreme-edge domain is exploited to automatically discover the available nodes, detect new nodes entering the clusters and monitor the status and the reachability of the existing ones. In this sense, the level of programmability can be considered at 100%.
Automation [Degree]: All the extreme-edge nodes are automatically discovered, and their behaviour is continuously tracked as input for the resource allocation algorithms.

KVIs
Trustworthiness: This is achieved through the high level of automation in the discovery and management of extreme-edge nodes.
Sustainability: This is increased through the automation of resource discovery, which limits the need for manual configurations.

Core Capabilities
Integrated intelligence: The scenario demonstrates the logic integrated into the orchestrator for the management of the extreme-edge resources for discovering, monitoring and dynamic allocation purposes.

Experiment 3: Usage of Simu5G

As reflected in Section 6.3, the main objective of this experiment is to analyse the impact of the Scenario 5.1 software components on a 5G/B5G RAN. This impact has been studied in terms of the number of resources consumed on the uplink and downlink and, besides, in terms of performance related to the network delay introduced experienced by each modelled SW component. Thereupon, below, a list with the KPIs/KVIs and Core Capabilities related to this scenario are given. Below are the KPIs, KVIs and Core Capabilities that are considered relevant in the scope of this experiment.

KPIs
Latency [s]: As mentioned, this KPI is used as the reference point in this experiment. The network delay experienced by each SW component is studied under different circumstances (i.e., the concentration of UEs) in order to understand the viability of deploying Scenario 5.1 components within a 5G/B5G RAN, considering its inherent network latency.

KVIs
Trustworthiness: It applies to this experiment regarding the availability and performance of the modelled SW components within a simulated 5G/B5G RAN.

Core Capabilities
Usage of Embedded Devices: The *sumoee* and the *tlctrl* SW component instances in this scenario are modelled to reflect the behaviour of those components when they are deployed on extreme-edge devices (i.e., Raspberry Pis). Therefore, although from a simulated point of view, it considers the integration of extreme-edge resources and, moreover, the integration of the *ee-edge-cloud continuum* as there are modelled devices on the various compute domains (e.g., extreme-edge, edge and cloud).
Flexibility: The extracted experiment results help to understand if this deployment is flexible enough to be added to a 5G/B5G environment. Flexibility can be envisioned in this experiment as a consequence of disaggregation and softwarisation of the involved SW components.

7.4 Lessons learnt

Implementation of demos has implied to use of multiple technologies, simulators and other tools. This process included the integration of multiple platforms. During experiments, the monitoring of some processes and components has been done. As a result of the implementation, the following experience has been gained:

Generic experience gained

- The usage of Open Source has been demonstrated to be crucial to validate new technological enablers such us the extreme-edge, distributed AI at the EDGE, data mesh implementation, and others. The challenge is to make these frameworks carrier-grade, to make them able to support telco workloads and the upcoming 6G use-cases.
- Experience in designing and developing a real-time monitoring tool for the extraction of metrics in Demo #4.

Continuum Orchestration related experience

- Learning techniques and tools to implement the unified orchestration across the “extreme-edge, edge, cloud” continuum, mainly regarding the integration of the extreme-edge scope in the M&O workflows. This has been explored in both Demos #4 and #5 and also in the “extreme-edge nodes discovery” lab experiment.
- Learning on requirements and limitations to connect infrastructure in separate networks and from different providers and domains under a common orchestration plane.
- Learning about designing and developing B5G microservices environment. This has been performed and validated in Scenario 5.1.
- Learning on integrating a simulated 5G RAN within Scenarios 5.1 and 5.2.

Extreme-edge related experience

- Platform development for arm64 architectures focused on low-power devices (Raspberry Pi specifically). The development of containers/pods for these devices requires specific developments based on arm64 architecture instead of amd64, as the typical bare metal servers.
- Studying the impact of the orchestration and monitoring actions on battery-powered extreme-edge nodes (experiment 6.2).
- Studied the impact of multithreading-based applications on extreme-edge containerised network functions. This has been explored in the context of Scenario 5.1.

AI/ML-related experience

- Learning on designing and developing orchestration mechanisms for provisioning, updating and re-configuring ML functions, combined with ML models' re-training and distribution, in the context of multi-domain scenarios involving the SW Vendor and the MNO scopes (addressed in Scenario 5.4). This particular structure has led to challenges that are not present in a typical MLOps approach, with just a single entity being the main ones:
  - Having some of the components exposed from one entity to the other so that both can access them (in line with the API Management Exposure concept).
  - Applying anonymization techniques to data before it is shared from one entity to another in order to avoid compromising privacy.
  - Identify in which environments it should be possible to make changes to the software package delivered by the SW Vendor to the MNO (and in which not) in order to anticipate the agreement between the two entities set out in the Service Level Agreement.
- Gaining experience in using AI/ML to predict short- and long-term variations in the deployed service that can effectively support the orchestration of edge resources and enable proactive approaches instead of reactive ones. It has also been seen that, in this context, a mixed simulation/emulation approach can provide valuable insights into evaluating the performance of the system at various scales.

Security related experience

- Learning on designing and integrating the Level of Trust Assessment Function (LoTAF) in the M&O architecture. This has been explored in the context of Scenario 5.3, where LoTAF has been presented as a Security Function together with the Security Closed Control Loops.
- Gained new knowledge in Physical Layer Security, a field in which the consortium had no prior experience.
- Improvements regarding the comprehension of various mechanisms involved in security, such as trust, automatization frameworks and cybersecurity frameworks.
- From a practical point of view, developing the security-related scenario in Demo #5 leads to searching, comparing, testing, deploying and developing tools to implement the cybersecurity frameworks and secure the user plane. This includes tools for monitoring, analysing, taking decisions and applying those decisions, intrusion protection systems, targeting vulnerable applications, using dedicated solutions to connect all those components, and management and orchestration tools such as OpenStack and K8s.

7.5 Future work

This deliverable includes a description of the implementation of two extensive demos, namely Demo #4 and Demo #5 and three complementary lab experiments. Many mechanisms of the Hexa-X M&O architecture, as described in previous subsections, have been evaluated. It doesn't mean that the evaluation is complete. In future work, we see the following topics:

a) More work should be performed regarding the specific tools and technologies that could be used to implement the proposed architecture and the possible standards to align with.
b) Some additional work on the API Management exposure. It includes:

- The alignment between the API Management Exposure concept and some reference solutions that could be used for implementing this concept, such as the Common API Framework for 3GPP northbound APIs (CAPIF) [CAP], should be checked.
- Evaluation of how the API Management Exposure Concept can be used in the context of the CAMARA project [CAM] to (i) make "service" APIs discoverable and consumable to targeted customers, including B2B and B2B2C customers, with optional aggregators (hyperscalers and CPaaS providers' marketplaces) mediating in between; and (ii) provide entry-points for operators to constitute federated environments.
- Possible integration of CAPIF framework for data and API exposure in MLOps scenarios to facilitate the secure and controlled exchange of data from the operator to the service provider domains and between staging and operational environments, as needed to guarantee meaningful datasets for ML training purposes.

c) All the innovations mentioned in D6.2 have been explored, except this one, i.e., intent-based networking that needs implementation and validation.
d) Investigating MLOps techniques for the deployment of other well-known ML paradigms (e.g., RL and FL) beyond the current approach based only on deploying supervised learning models. Explore also other anonymization and encryption techniques for the showcased supervised learning model (or other similar models). Also, considering other, more complex multi-vendor approaches regarding the MLOps approach, towards the increasing level of automation, targeting full zero-touch automation (ZTA) whenever possible.
e) More research regarding the extreme-edge nodes discovery and the integration of the radio.
f) Improving resource orchestration of extreme-edge nodes in end-to-end scenarios, e.g., considering the impact of their mobile connectivity on the management interactions between platform and worker nodes, enhancing the resource allocation logic with constraints related to per-node characterization, and using ML techniques to predict the time-variable attributes.
g) Coordinate management actions for data collection/transfer/storage and ML pipeline automation in distributed and multi-domain environments, taking into account data characteristics (ownership, sharing policies, privacy, etc.), also associated with MLOps scenarios.
h) Coordination of automation and closed-loop decisions and actions across multiple domains, infrastructure layers and time scales.
i) Evaluation of a large-scale orchestration of different types of edge resources, having diverse resource capacity, performance, operation cost and availability, and providing services to heterogeneous user applications. To analyse the impact of the prediction algorithm and forecast (with the corresponding uncertainty) on performance.
j) Using localization information, possibly linked to the node discovery functionality, to enhance the operations of resources and network functions placement.
k) Extending the reach of the performance diagnosis and functions placement mechanisms, shown in Scenario 4.2, from the services all the way to the network and the M&O components as well.
l) Further study on 'network-of-networks', coordinating the monitoring and control actions upon flexible topologies/networks, functions placement, and unified orchestration for the purpose of being able to provide ad-hoc networks to new *orchestratable* resources on-demand.
m) Further research, design and development in the two stages of the Level of Trust Assessment Function (LoTAF). Scenario 5.3 spotlights some high-level characteristics of the second LoTAF stage. Nevertheless, LoTAF also encompasses management and automation actions, such as network service selection based on user's security and privacy requirements, intelligent optimization functions or technology-based threat analysis, which are considered interesting for future work.

8 Conclusion

This report, as the final deliverable of the Hexa-X Work Package 6 (WP6), evaluates service management and orchestration (M&O) mechanisms for Hexa-X, described in Hexa-X D6.2 [HEX22-D62]. It introduces the implementation of novelties described in D6.2, such as (1) unified orchestration across the "extreme-edge, edge, core" continuum, (2) unified management and orchestration across multiple domains owned and administered by different stakeholders, (3) increasing levels of automation, (4) adoption of data-driven and AI/ML techniques in the M&O system, (5) adoption of the cloud-native principles in the telco-grade environment.

This deliverable's main part consists of describing two demos (Demo #4, Demo #5) and other complementary lab experiments. Both demos have addressed Hexa-X Objective 3 (Connecting intelligence towards 6G). Specifically, Demo #4 was focused on unified management and control using Cloud – Edge – Extreme-edge continuum orchestration on a Digital Twins service (Scenario 4.1), handling unexpected events using dynamic functions placement (Scenario 4.2), and improving service downtime and reducing costs using predictive orchestration (Scenario 4.3). On the other hand, Demo #5 has implemented and evaluated AI/ML-driven operations supported by continuum orchestration (Scenario 5.1), prediction-based service orchestration and optimization (Scenario 5.2), reactive security for the edge (Scenario 5.3), and the application of MLOps techniques to deploy AI/ML service components (Scenario 5.4). Other lab experiments have aimed to address network energy efficiency, extreme-edge nodes discovery as well as the impact of the RAN on Scenario 5.1. Both demos have been presented in the document in a uniform fashion, consisting of (i) Demo overview, (ii) Innovations related to the demo and (iii) Demo implementation, of which the last one additionally describes individual scenarios related to each particular demo. All of the scenarios are again presented in a uniform fashion, consisting of (a) scenario description, (b) software components, (c) functional behaviour and (d) deployment.

In the second main part of this document, the evaluation of proposed service management and orchestration mechanisms has been performed. The evaluation includes a description of the WP6 contribution to the overall Hexa-X objectives, focusing on Objective 3 and considering the M&O-related topics linked to that Objective. This includes the main WP6 output towards such Objective 3, the Objective 3 measurable results and the WP6-related quantifiable targets. The evaluation also includes the validation of the Hexa-X M&O architectural design provided in the previous Deliverable D6.2 [HEX22-D62], the analysis of the main KPIs, KVIs and Core Capabilities related to the demos and the lab experiments, and provides the main lesson learnt and some hints and suggestions for future work. The evaluations show that the assumed objectives have been achieved.

Annex I. SUMO-related implementation details

Figure AI - 1 shows the SUMO vehicular topology that has been generated for Scenario 5.1. As can be seen, there is a combination of simple roads (i.e., with one single lane per direction) and complex roads (i.e., with two lanes per direction).

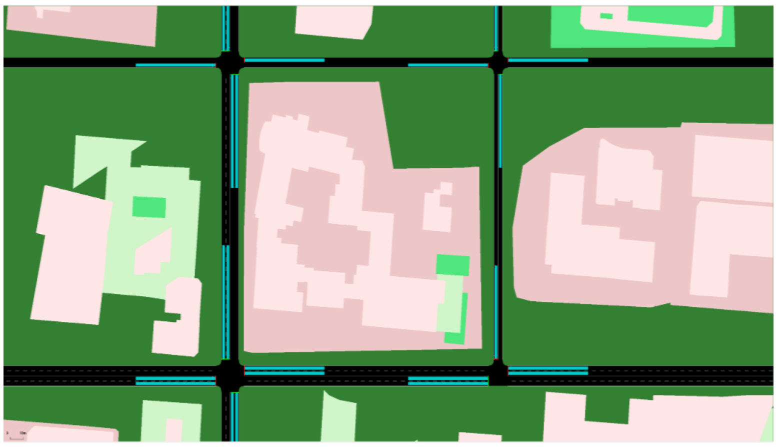


Figure AI - 1. Scenario 5.1 SUMO layout.

In order to reflect real-life road conditions, each road has been configured with different maximum speeds and lengths. As reflected in Figure AI - 2, the south horizontal roads and the west vertical roads have a higher maximum admitted speed and are comprised of two lanes, while the north horizontal roads and the east vertical roads have been configured with a much lower maximum speed and as simple roads. The idea behind this design is to simulate two main roads with higher traffic volumes (e.g., roads coming from a highway exit that enter a city, deviation roads towards a highway, etc.) and also secondary roads that may pose an alternative route to reach a destination although they have lower speed and a higher probability of getting a traffic jam (e.g., city centre roads or secondary roads within the old town of a city). Additionally, the roads have been given different lengths to allow the vehicles that enter and exit the SUMO scenario to reach their maximum allowed speeds in zero traffic jam conditions.

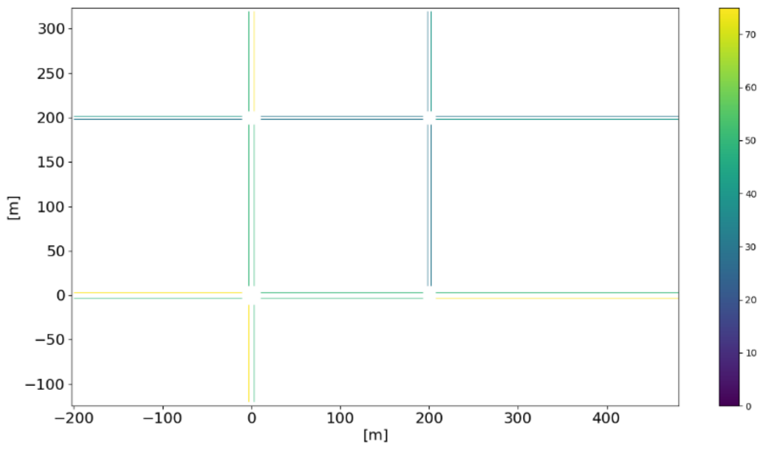


Figure AI - 2. Scenario 5.1. SUMO road speeds and length.

On the other hand, SUMO allows its users to add traffic light objects to each existing crossroad on the designed vehicular topology [LBE+18]. From the SUMO perspective, there is only one traffic light object controlling the crossroad (see Figure AI - 3); however, this SUMO traffic light object is able to manage each lane that comes into the crossroad as an independent traffic light thanks to a tuple of parameters that simulate a per-lane traffic lane granular behaviour for the traffic light phases[15]: *(i) Duration,* indicates the duration, in seconds, of a traffic light phase state; *(ii) State*, indicates to the SUMO traffic light object the state on each individual, per-lane and per-destination, traffic light status (e.g., green, yellow or red) as a simple string. To further clarify this point, as it is key for the implementation of this scenario, Figure AI - 3 represents the SUMO traffic light object of the south-west crossroad represented in Figure AI - 1, which details how a traffic light object is interpreted by SUMO. As it can be seen, for each lane entering the crossroad, all the possible paths are coloured, reflecting the state of the potential independent traffic light that serves that path. For instance, the incoming northern lanes have four potential destinations, i.e., going forward on each lane, turning to left and turning to right; each individual path can be controlled as if a dedicated traffic light was serving that specific path. The *State* string that would reflect the traffic light state in Figure AI - 3 will be *"GGGgrrrrGGGgrrrr"*[16]; from the beginning of the string, the four first characters reflect the state of each individual path, from left to right, of the northern road. Then the following four characters reflect the state of each path of the eastern road, and the same pattern applies to the southern and western roads, respectively. The opposite traffic light state (green on the horizontal lanes and red on the verticals) will be achieved with the following traffic light state string *"rrrrGGGgrrrrGGGg"*.

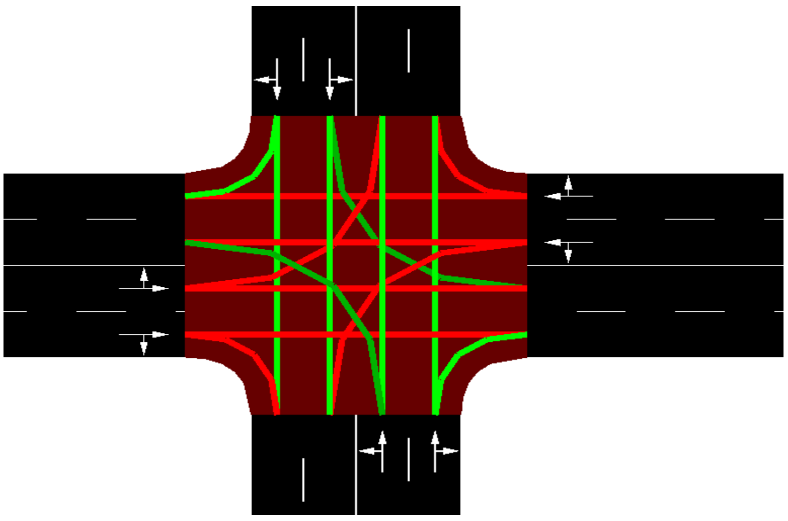


Figure AI - 3. Scenario 5.1 - SUMO TL object view.

The SUMO TraCI API [LBE+18] enables different ways of managing each SUMO TL object when the simulation is running. The most relevant ones are listed below:

1. **State Change**: This method sets the TL state to the string passed to the API. After this method is used to change a TL state, that TL will be set to "*online*", and the state will remain as it is until the next to this method is received or a new program (i.e., a full set of TL phases – states plus the duration of each state) is loaded to the TL.
2. **Phase duration Change**: Allows to set the remaining duration of the current TL phase to the desired value.
3. **Full traffic lights program load:** Inserts a completely new traffic lights program into the desired traffic light. This means that each phase is edited in duration and state.

In this scenario, the first method has been selected as it allows the *AI/RL Agent* functional component more freedom due to the fact that the state and the duration of a TL can be both managed with a single command and with high granularity. Besides, as the available GPIO ports on each Raspberry are limited, which means that the traffic light that can be emulated using

[15] In SUMO, a traffic light phase can be defined as the time that a traffic light holds the same light state.

[16] SUMO enables even more granular control of the traffic lights state by defining two types of "*Green*" states: one (represented with a capital "G") means that vehicles following that path have the priority, and another one (represented with "*g*"), meaning that vehicles may pass the junction if no vehicle is with a higher priority, otherwise they decelerate for letting it pass.

physical LEDs is also limited, for this scenario, a single traffic light is considered for each crossroad, independently of the number of lanes and possible paths that this road may have towards other roads, i.e., in Figure AI - 3's crossroad there are four traffic lights instead of the 16 that SUMO enables. Moreover, SUMO comes with various default vehicle types, but they are limited and therefore, to make the simulation more realistic, custom vehicle types were implemented for this scenario. A vehicle type in SUMO allows the user to define the features that define the vehicle during the simulation: vehicle length/width, maximum speed, acceleration/deceleration, shape, colour, consumption model, emission model, driver's driving imperfection, etc[17]. Table AI - 1 shows the features of each custom vehicle type.

Table AI - 1. Scenario 5.1: SUMO vehicle types.

Type	**Class**	**Length (m)**	**Colour**	**Max. Speed (m/s)**	**Accelera-tion (m/s^2)**	**Decelera-tion (m/s^2)**	**Sigma [18]**	**Emission Model[19]**
CarA	Car	5.0	Yellow	60	3.0	6.0	0.6	PC_G_EU4
CarB	Car	6.5	Blue	50	2.4	5.0	0.5	PC_D_EU4
CarC	Car	4.5	Green	40	1.5	4.5	0.3	PC_D_EU1
MbikeA	Motor-cycle	1.8	Red	55	3.0	5.0	0.6	LDV_G_EU4
Emer-gency A	Emer-gency	6.0	White	50	2.0	6.5	0.5	HDV

Several vehicle emission models have been implemented with the objective of demonstrating how much the *AI/RL agent* functional component is able to optimize their respective consumptions. Besides, apart from the physical parameters of the vehicle types, different driving behaviours have been associated with each type of vehicle. For example, *CarA* vehicle type represents high-end vehicles, and it has been given a sigma of 0.6, while the low-end vehicle type, *CarC*, has been given a sigma of 0.3. Furthermore, SUMO requires vehicular routes to be defined in order to allow traffic displacements to occur during the simulation execution. Again, for the purpose of replicating a real-life scenario, all the possible routes from any existing junction to any other junction have been enabled in this SUMO simulation. Nonetheless, the traffic flows have been implemented in such a way that, at the start of the simulation, there is a light traffic volume, but as the simulation continues, more and more traffic starts to appear. This approach tries to reflect a vehicular situation where a traffic jam occurs when people start ending their workday and begin returning home. Finally, in order to simulate the field of vision of a camera attached to each road junction traffic light, a SUMO Lane Area Detector (LAD) object has been added to each lane on each road at the nearest point to the crossroad (see blue rectangles in Figure AI - 1). These objects cover a certain area and act as a geofence where different data from the vehicles within that area can be retrieved. All the LADs implemented for this scenario have a length of *m* in order to enable more than five vehicles, on average, to enter the same LAD.

[17] See [SUMV22] for a detailed view of the available vehicles' customizable parameters.

[18] Driving imperfection: 0 equals to perfect driving and 1 worst possible driver.

[19] [SUME22]

Annex II. RL Agent implementation details

The AI/ML application developed for the demonstration in Scenario 5.1 consists of four Reinforcement Learning agents which deal with the traffic lights control by changing its state after some simulation time steps. As introduced in Section 5.3.1.2, the agents implement the Q-Learning algorithm [Wat89], a model-free algorithm that aims to learn the best *actions* to be performed in a defined environment (the urban environment with the traffic lights in the context of this demo) depending on the *rewards* it receives to the action it performs. Below are some more details about how the environment state is defined, how the rewards are computed, and how the actions are encoded:

A) State: The state each agent receives contains the information of all the vehicles detected by the Lane Area Detectors (LADs) from the SUMO simulator. As can be seen in Figure 5-3, they have been defined on the traffic simulation topology, each one retrieving information about the number of vehicles in the respective LAD area. Specifically, the state for each LAD is a binary value (0 or 1), indicating if the number of vehicles in the LAD is above or below a certain threshold to represent traffic jams or smooth traffic situations. Putting together all these values, a 15-bits string is built-up, representing the whole state of the road traffic nearby all the traffic lights in the set-up. This 15-bit string is the *state* information sent to each agent.

B) Action: Action commands are encoded per crossroad, considering that there are two different kinds: Type A (with four traffic lights – three of them) and Type B (with only three traffic lights – one of them, see Figure 5-3). Considering this and also discarding certain actions that are not desirable (e.g., putting all traffic lights on green or red at the same time), the following sets of actions have been defined per type of crossover[20]:

C) Reward: Firstly, a base score is computed according to the following formula for each LAD:

Table AII - 1. Actions for crossroad types A (light) and B (dark).

	Traffic Light North	Traffic Light East	Traffic Light West	Traffic Light South
Action 1	Green	Red	Red	Green
Action 2	Red	Green	Green	Red
Action 3	Green	Red	Red	Red
Action 4	Red	Green	Red	Red
Action 5	Red	Red	Green	Red
Action 6	Red	Red	Red	Green

According to this table, actions on each crossroad are encoded as 3 or 4 bits numbers for crossroads of type B or A respectively, e.g., Action 2 for a crossroad type A would be encoded as red-green-green-red (or 0110) while for the crossroad type be it would be red-green-green (or 011). This would be the output from each RL Agent towards the Traffic Lights Control Logic module.

D) Reward: Firstly, a base score is computed according to the following formula for each LAD:

$$reward = \left(\overline{vehicles\_speed} - \overline{vehicles\_waiting\_time}\right)\alpha$$

Where $\overline{vehicles\_speed}$ is the mean speed of the vehicles, $\overline{vehicles\_waiting\_time}$ is their mean waiting time, and α is a constant factor intended to prevent the score value from swinging too wide (the concrete value has been obtained experimentally on the simulation). The waiting time and speed variables have been used to calculate the score in order to give a higher result if the vehicles speed

[20] Note the yellow colour is not indicated here. This is because it is only considered a transition state, which duration is directly controlled by the Traffic Lights Control Logic module.

in a certain LAD is high, which indicates that the fluidity is good. On the contrary, the score is lower or even negative if the vehicles are waiting so much time in a certain LAD, which indicates that the traffic fluidity is bad. To perform the learning process, the AI/ML component works in a synchronised way. It waits until the state information from all the crossroads is available at its input. Then, once the data is collected, the AI/ML component process the data to compute the actions on the different crossroads to the current iteration (i.e., the current state) and the rewards for the actions that were performed in the previous iteration.

The reinforcement learning algorithm takes the decisions by using a so-called Q-learning table (where 'Q' comes from 'quality'), where the so-called Q-values are stored for each state-action pair. This table is structured as follows: the first column (the table key) represents the state, with the 15-bits word mentioned above. Then, the remaining columns represent the possible actions for each crossroad, according to those in the previous Table AII - 1. The values stored in the table for each row (each state) represent how positive was the effect of applying each specific action to that state. The higher the value, the higher is considered the "quality" of the action. Table AII - 2 shows the layout of this Q-learning table, where columns A_1 … A_6 represent the possible actions in Table II - 1. To select the specific action for each crossroad, each agent selects from the table the action with the highest Q-value (e.g., in the 1st row, for the 1st crossroad, action A_6 would be chosen over A_1).

Table AII - 2. Q-table layout.

	Crossroad 1			Crossroad 2			…	Crossroad 4		
State	A_1	...	A_6	A_1	...	A_6	…	A_1	...	A_4
001101001101000	7.2101	...	9.9104	1.1132	...	1.1211	...	1.1143	...	2.1051
111010001100000	0.9112	...	1.1111	5.1059	...	9.1052	...	4.3110	...	0.0
...	...	...	...	...	...	...	...	...	...	...

The Q-values are updated according to the following formula:

$$q^{new}(s,a) = (1-\alpha)q(s,a) + \alpha(R_{t+1} + \gamma \max_{a'} q(s',a'))$$

where

- $q^{new}(s,a)$ the new q value that will be updated on the table for a specific state-action pair.
- $q(s,a)$ the previous q value in the table (if any).
- α, the so-called *learning rate*. It regulates how the Q-Value calculated in the previous time step is going to affect over the new computed Q-Value. It is basically used to avoid overwriting the previous Q-Value. The learning rate can be set between 0 and 1, the more close to one, the faster the agent will adopt the new Q-Value and vice versa.
- γ, the discount rate. Set the importance of future rewards. A rate of 0 will make the agent to only consider the current rewards. While a rate close to 1 will take into consideration the previous ones.
- R_{t+1} the reward to the next iteration.
- $\max_{a'} q(s',a')$, the optimal Q-Value for the next state-action pair. In practice, the maximum q-value in the table for the next state.

Additionally, it should be remarked that since it takes some time for the actions to have a significant effect on the road traffic, it has been done that one iteration for the agents is equivalent to several simulation time steps. This makes reward values more significant. When the simulation starts, the table is initially empty, i.e., with no row. New rows are added while the simulation carries on, and new states are detected in the environment. It is worth mentioning that, although there are a maximum of 32,768 possible states (2^{15}, considering the 15 LADs), during different demo executions, it has been seen that a quite good improvement in the road traffic quality was achieved once the table reached approximately 500 entries. In other words, RL Agents learned that, out of the 32.768 possible states, it was possible to manage the traffic already in an efficient way considering only about 500 states. Annex III shows the results in that situation, i.e., the learning process until the RL Agents had learned about those 500 states approximately.

Annex III. Results of the RL-based traffic lights control service

This annex shows some of the results obtained in the AI/ML-driven traffic lights control service deployed in Scenario 5.1. Although, as said, the main interest of this scenario is not the AI/ML part itself, it is considered interesting to show here how the service behaves, even in summary form. The plots below show in a comparative way some interesting KPIs obtained through the SUMO simulator using the demo set-up in Figure 5-13. In all cases, the green line represents the results using the Reinforcement Learning approach to control the traffic lights activation, while the black line represents the "legacy" approach, i.e., the traffic lights control using the regular fixed-time pattern that is typically used in real-life cities. It is important to remark that this legacy behaviour is based on the timers defined by the SUMO simulator itself, i.e., not by the team in charge of executing the simulation. These timer values are automatically generated by SUMO considering different aspects (such as the type of crossroads and the number of lanes per road, among others [SUME22]), and estimating the values these timers could have in a similar real-life scenario. As can be seen, the following metrics are plotted:

- Total vehicles over time (Figure AIII - 1). It represents the number of vehicles that were running within the reported simulation time step.
- Halting vehicles over time (Figure AIII - 2). It represents the number of vehicles in the network with speed below 0.1m/s for a given simulation time step.
- Vehicles mean travel time (Figure AIII - 3). It represents the mean travel time of all vehicles that have left the simulation within the previous and the reported time.
- Vehicles mean waiting time (Figure AIII - 4). It represents the mean waiting time for all vehicles, up to the given simulation time step and within the reported time step, in order to be inserted into the simulation.
- Cumulative emitted hydrocarbons (Figure AIII - 5). It represents the accumulated amount of hydrocarbons emitted by all the vehicles up to the reported simulation time step.
- Cumulative fuel consumption (Figure AIII - 6). It represents the accumulated amount of fuel consumed by all the vehicles up to the reported simulation time step.

Cumulative emitted CO_2 (

- Figure AIII - 7). It represents the accumulated amount of CO_2 emitted by all the vehicles up to the reported simulation time step.

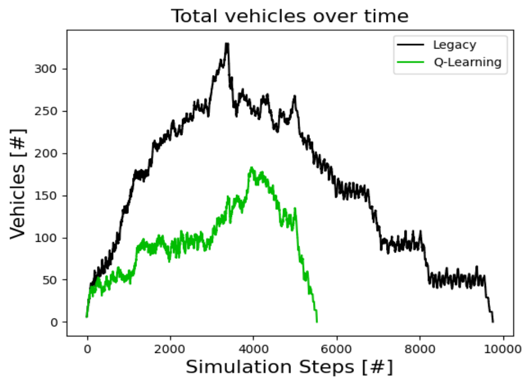


Figure AIII - 1. Scenario 5.1 - simulation total vehicles comparative.

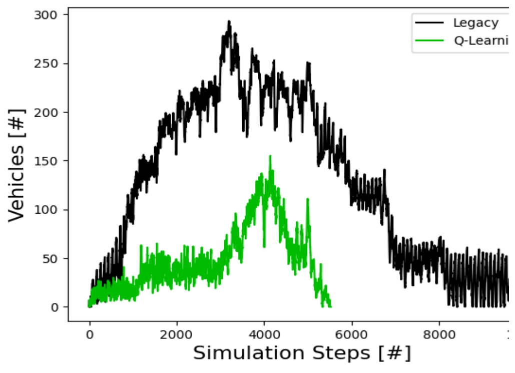


Figure AIII - 2. Scenario 5.1 - simulation halting vehicles comparative.

As it can be seen, for all the metrics, the RL-based approach shows better results than the legacy approach, being very significant vehicle waiting times (much longer in the legacy approach), and also, the graphs related to gas emissions and vehicles consumption (lower when using the RL approach). Note also that, in all cases, the *RL-based* approach (green line) ends much sooner than the *Legacy* approach (almost 5000 steps sooner). This is because, in both cases, the number of vehicles generated by SUMO is the same, so this basically shows how the RL-based approach gets all the vehicles in the simulation to complete their journey in a shorter time (the condition for the simulation to end is that all the vehicles have reached their expected destination). Thereupon, it can be concluded that the *RL-approach* greatly optimizes the overall traffic flows.

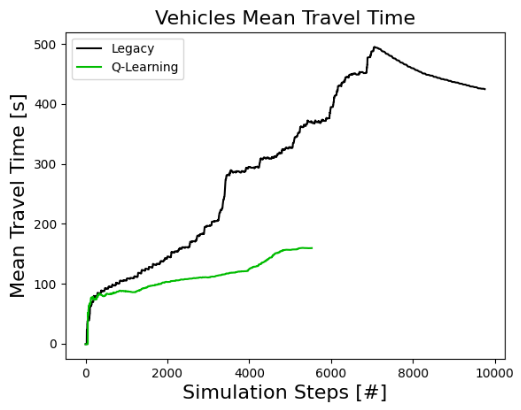


Figure AIII - 3. Scenario 5.1 - simulation vehicles mean travel time comparative.

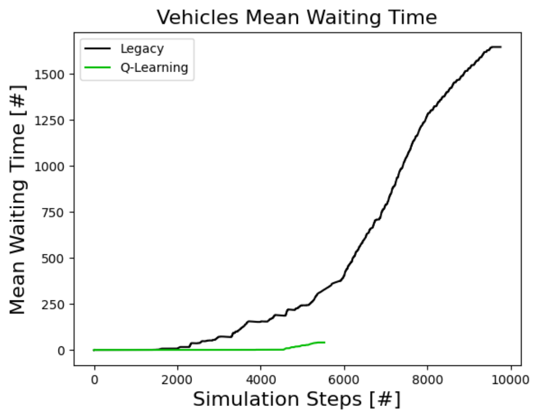


Figure AIII - 4. Scenario 5.1 - simulation vehicles mean waiting time comparative.

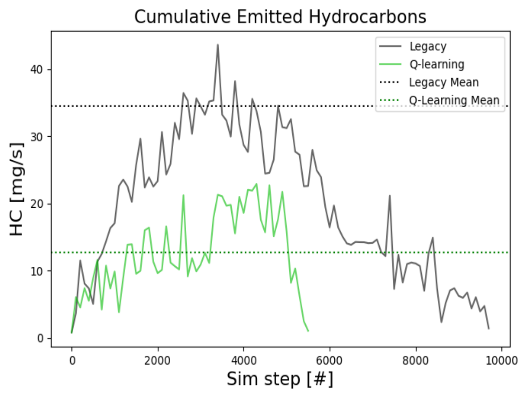


Figure AIII - 5. Scenario 5.1 - simulation vehicles HC comparative.

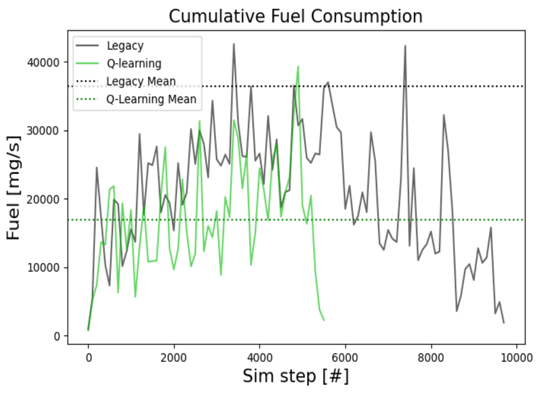


Figure AIII - 6. Scenario 5.1 - simulation vehicles fuel comparative.

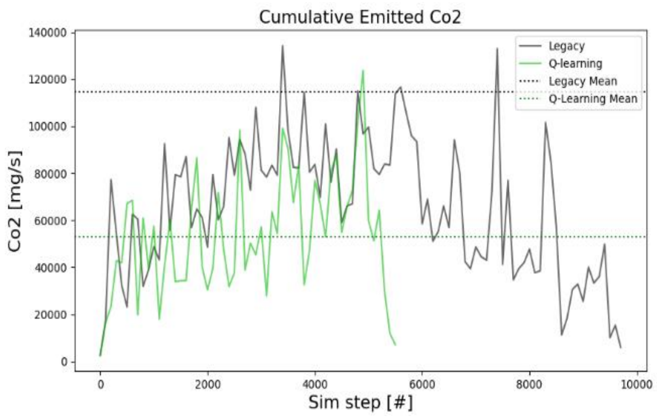


Figure AIII - 7. Scenario 5.1 - simulation vehicles CO_2 comparative.

References

[5GA21] 5G Americas, “Vehicular connectivity: C-V2X & 5G,” White Paper, September 2021.

[5GR21-D24] 5Growth Deliverable D2.4, “Final implementation of 5G End-to-End Service Platform”, May 2021, [Online]. Available at: https://5growth.eu/wp-content/uploads/2019/06/D2.4-Final_implementation_of_5G_End-to-End_Service_Platform.pdf [Accessed: 1 March 2023].

[AIR] Apache Airflow [Online]. Available at: https://airflow.apache.org/ [Accessed: 28 February 2023].

[ARI] ARIMA, [Online]. Available at: https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average [Accessed: 24 February 2023].

[BBE+11] M. Behrisch, L. Bieker, J. Erdmann and D. Krajzewicz, “SUMO–simulation of urban mobility: an overview.” Proceedings of SIMUL, The Third International Conference on Advances in System Simulation, 2011.

[BPV+22] F. Barbarulo, C. Puliafito, A. Virdis, E. Mingozzi, "Enabling Application Relocation in ETSI MEC: A Container-Migration Approach", International Workshop on Cloud Technologies and Energy Efficiency in Mobile Communication Networks (CLEEN) 2022, Virtual Conference, 12–15 September 2022.

[CAP] Common API Framework, [Online]. Available at: https://www.mpirical.com/glossary/capif-common-api-framework-for-3gpp-northbound-apis [Accessed: 27 February 2023].

[CAM] Camara Project, [Online]. Available at: https://camaraproject.org [Accessed: 27 February 2023]

[CAR+03] M. Carson and D. Santay, "NIST Net: A Linux-based network emulation tool", ACM SIGCOMM Comput. Commun. Rev., vol. 33, no. 3, pp. 111-126, Jul. 2003.

[CVE-L4J] CVE-2021-44228, [Online]. Available at: https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2021-44228 [Accessed: 27 February 2023].

[COR] Log4jHotPatch, [Online]. Available at: https://github.com/corretto/hotpatch-for-apache-log4j2 [Accessed: 6 March 2023].

[DBS] DBSCAN, [Online]. Available at: https://en.wikipedia.org/wiki/DBSCAN [Accessed: 27 February 2023].

[DOC] Docker, [Online]. Available at: https://www.docker.com/ [Accessed: 8 March 2023].

[DPE] Dell Power Edge, [Online]. Available at: https://www.dell.com/es-es/shop/servidores-dell-poweredge/sc/servers [Accessed: 21 March 2023].

[DRO] Drools, [Online]. Available at: https://drools.org [Accessed: 24 February 2023].

[Ela19] Elasticsearch B.V. 2019. Elasticsearch Beats - Lightweight Data Shippers. [Online]. Available at: https://www.elastic.co/products/beats/. [Accessed 15 May 2019].

[EAD14] Erich F., Amrit C., Daneva M., “A Mapping Study on Cooperation between Information System Development and Operations”, In: Jedlitschka A., Kuvaja P., Kuhrmann M., Männistö T., Münch J., Raatikainen M. (eds) Product-Focused Software Process Improvement. PROFES 2014. Lecture Notes in Computer Science, vol 8892. Springer, Cham, 2014.

[FPR] Facebook Prophet, [Online]. Available at: https://facebook.github.io/prophet/ [Accessed: 24 February 2023].

[GDH+13].	Anshul Gandhi, Sherwin Doroudi, Mor Harchol-Balter, and Alan Scheller-Wolf. 2013. Exact analysis of the M/M/k/setup class of Markov chains via recursive renewal reward. SIGMETRICS Perform. Eval. Rev. 41, 1 (June 2013), 153–166. https://doi.org/10.1145/2494232.2465760
[GDPR]	GDPR, [Online]. Available at: https://gdpr-info.eu/ [Accessed: 8 March 2023].
[GIT]	GitLab, [Online]. Available at: https://about.gitlab.com/ [Accessed: 24 February 2023].[GRA] Grafana, [Online]. Available at: https://grafana.com/ [Accessed: 8 March 2023].
[HELM]	Helm Charts, [Online]. Available at: https://helm.sh/docs/topics/charts/ [Accessed: 24 February 2023]
[HEX21-D12]	Hexa-X Deliverable D1.2: Expanded 6G vision, use cases and societal values – including aspects of sustainability, security and spectrum. April 2021.
[HEX22-D14]	Hexa-X Deliverable D1.4: Hexa-X architecture for B5G/6G networks. April 2023.
[HEX23-D43]	Hexa-X Deliverable D4.3, "AI-driven communication & computation co-design: final solutions", [Ongoing Work].
[HEX21-D61]	Hexa-X Deliverable D6.1: Gaps, features and enablers for B5G/6G service management and orchestration. June 2021.
[HEX22-D62]	Hexa-X Deliverable D6.2: Design of service management and orchestration functionalities. April 2022.
[HEX22-D72]	Hexa-X Deliverable D7.2: Special-purpose functionalities: intermediate solutions. April 2022.
[Hor20]	Horizon 2020 Work Programme 2016–2017 [Online]. Available at: https://ec.europa.eu/research/participants/data/ref/h2020/other/wp/2016-2017/annexes/h2020-wp1617-annex-ga_en.pdf [Accessed: 22 November 2022].
[INE]	INET Library Website. [Online]. Available at: https://inet.omnetpp.org [Accessed: 24 February 2023].
[INF]	InfluxDB, [Online]. Available at: https://www.influxdata.com/. [Accessed: 24 February 2023].
[IPE]	IPERF, [Online]. Available at: https://iperf.fr/. [Accessed: 27 February 2023].
[ISO2011]	ISO. 2011. ISO/IEC TS 25010:2011(en) Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. ISO. International Organization for Standardization, Geneva, Switzerland.
[ISO2017]	ISO. 2017. ISO/IEC TS 25011:2017(en) Information technology — Systems and software Quality Requirements and Evaluation (SQuaRE) — Service quality models. ISO. International Organization for Standardization, Geneva, Switzerland.
[IST]	Istio. [Online]. Available at: https://istio.io/ [Accessed: 8 March 2023].
[K3D]	K3d, [Online]. Available at: k3d.io/ [Accessed: 8 March 2023].
[K3S]	K3s, [Online]. Available at: https://k3s.io/ [Accessed: 27 February 2023].
[KUBa]	Kubernetes, [Online]. Available at: https://kubernetes.io/. [Accessed: 27 February 2023].
[KUBb]	Kubeflow, [Online]. Available at: https://www.kubeflow.org/. [Accessed: 24 February 2023].
[KUP]	Kubeflow Pipelines, [Online]. Available at: https://www.kubeflow.org/docs/components/pipelines/v2/introduction/. [Accessed: 27 February 2023].
[KAF]	Apache kafka [Online]. Available at: https://kafka.apache.org. [Accessed: 24 February 2023].

[L4J]	Apache log4j, [Online]. Available at: https://logging.apache.org/log4j/2.x/security.html [Accessed: 24 February 2023].
[LBE+18]	P.A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.P. Flötteröd, R. Hilbrich, and E. Wießner, “Microscopic traffic simulation using sumo.” 21st international conference on intelligent transportation systems (ITSC), pp. 2575-2582, IEEE, 2018.
[Ler17]	Andrew Lerner, AIOps Platforms, August 09, 2017 [Online]. Available at: https://blogs.gartner.com/andrew-lerner/2017/08/09/aiops-platforms. [Accessed: 24 February 2023].
[LST]	Long Short-term Memory, [Online]. Available at: https://en.wikipedia.org /wiki/Long_short-term_memory [Accessed: 6 March 2023].
[MAH+04]	D. Mahrenholz and S. Ivanov, "Real-time network emulation with ns-2", Proc. 8th IEEE Int. Symp. Distrib. Simulation Real-Time Appl., pp. 29-36, Oct. 2004.
[MKV+22]	J. Martín-Pérez et al., "Dimensioning V2N Services in 5G Networks Through Forecast-Based Scaling," in IEEE Access, vol. 10, pp. 9587-9602, 2022, doi: 10.1109/ACCESS.2022.3142346.
[MLR]	Multiple Linear Regression, [Online]. Available at: https://en.wikiversity.org/wiki/Multiple_linear_regression [Accessed: 27 February 2023].
[MIN]	MinIO, [Online]. Available at: https://min.io/. [Accessed: 24 February 2023].
[NGP18-D31]	NGPaaS Deliverable D3.1, Initial Dev-for-Operations Model Specification, April 2018, [Online]. Available at: http://ngpaas.eu/wp-content/uploads/2018/07/NGPaaS_D3.1_Web.pdf
[NGP19-D32]	NGPaaS Deliverable D3.2, Final Dev-for-Operations Model Specification, April 2019.
[NSS+20]	G. Nardini, D. Sabella, G. Stea, P. Thakkar, and A. Virdis, ‘‘Simu5G— An OMNeT++ library for end-to-end performance evaluation of 5G networks,’’ IEEE Access, vol. 8, pp. 181176–181191, 2020, doi: 10.1109/ACCESS.2020.3028550.
[OSM]	Open Source MANO, [Online]. Available at: https://osm.etsi.org/ . [Accessed:28 February 2023].
[OST]	OpenStack, [Online]. Available at: https://www.openstack.org/. [Accessed: 28 February 2023].
[PKH+19]	Palmer, G. I., Knight, V. A., Harper, P. R., & Hawa, A. L. (2019). Ciw: An open-source discrete event simulation library. Journal of Simulation, 13(1), 68-82.
[PAE]	Prometheus Authors. Exporters and Integrations. [Online]. Available at: https://prometheus.io/docs/instrumenting/exporters/ [Accessed: 24 February 2023].
[QUEC]	Quectel RM500Q mode, [Online]. Available at: https://www.quectel.com/product/5g-rm50xq-series [Accessed: 24 February 2023].
[RAB]	RabbitMQ, [Online]. Available at: https://www.rabbitmq.com [Accessed 8 March 2023].
[RCA]	Bhattacharya, Krishanu & Rani, N & Gonsalves, Timothy & Murthy, Hema. (2005). AN EFFICIENT ALGORITHM FOR ROOT CAUSE ANALYSIS.
[RF18]	J. Redmon, A. Farhadi, 2018 “YOLOv3: An Incremental Improvement”.
[ROS]	Robot Operating System, [Online]. Available at: https://www.ros.org/ [Accessed: 8 March 2023]

[SOM]	Kohonen, Teuvo (1982). <<Self-Organized Formation of Topologically Correct Feature Maps>>. Biological Cybernetics 43 (1): 56-69.
[SOM2]	Root-cause analysis through machine learning in the cloud, Tim Josefsson, [Online]. Available at: https://uu.diva-portal.org/smash/get/diva2:1178780/FULLTEXT01.pdf [Accessed: 8 March 2023]
[SRO+19]	R. C. Staudemeyer, E. R. Morris, Understanding LSTM -- a tutorial into Long Short-Term Memory Recurrent Neural Networks, September 2019, DOI 10.48550/arXiv.1909.09586 [Online]. Available at: https://arxiv.org/abs/1909.09586 [Accessed: 8 March 2023].
[STO+20]	N. Senavirathne, V. Torra, On the Role of Data Anonymization in Machine Learning Privacy, IEEE Access, December 2020, DOI 10.1109/TrustCom50675.2020.00093 [Online] available at: https://ieeexplore.ieee.org/abstract/document/9343198 [Accessed: 8 March 2023].
[SUME22]	SUMO Docs, “HBEFA3-based emission models - SUMO Documentation”, Accessed: Feb. 2023. [Online]. Available at: https://sumo.dlr.de/docs/Models/Emissions/HBEFA3-based.html [Accessed: 8 March 2023].
[SUMV22]	SUMO Docs, “Definition of Vehicles, Vehicle Types, and Routes - SUMO Documentation”, Accessed: Feb. 2023. [Online]. Available at: https://sumo.dlr.de/docs/Definition_of_Vehicles%2C_Vehicle_Types%2C_and_Routes.html#vehicle_types [Accessed: 8 March 2023].
[SUR]	Suricata, [Online]. Available at: https://suricata.io/ [Accessed: 24 February 2023].
[TBF+22]	M. Testi, M. Ballabio, E. Frontoni, et al, MLOps: A Taxonomy and a Methodology, IEEE Access, June 2022, DOI 10.1109/ACCESS.2022.3181730, [Online]. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9792270 [Accessed: 8 March 2023].
[TFDV]	TensorFlow Data Validation, [Online]. Available at: https://www.tensorflow.org/tfx/guide/tfdv [Accessed: 28 February 2023].
[TFL]	TensorFlow, [Online]. Available at: https://www.tensorflow.org/. [Accessed: 27 February 2023].
[TFMA]	TensorFlow Model Analysis, [Online]. Available at: https://www.tensorflow.org/tfx/guide/tfma [Accessed: 28 February 2023].
[TFS]	TensorFlow, “TensorFlow Serving” [Online]. Available at: https://www.tensorflow.org/tfx/guide/serving [Accessed: 24 February 2023].
[TFX]	TensorFlow Extended, [Online]. Available at: https://www.tensorflow.org/tfx. [Accessed: 24 February 2023].
[TR873]	3GPP TR 36.873 v12.7.0, "Study on 3D channel model for LTE", December 2017.
[TRI]	Trivy, [Online]. Available at: https://aquasecurity.github.io/trivy/v0.28.1 [Accessed: 26 April 2023].
[TRL]	Technology Readiness Levels, [Online]. Available at: https://ec.europa.eu/research/participants/data/ref/h2020/wp/2014_2015/annexes/h2020-wp1415-annex-g-trl_en.pdf [Accessed: 27 February 2023].
[UER]	UERANSIM, [Online]. Available at: https://www.free5gc.org/installations/stage-3-sim-install/ [Accessed: 24 February 2023].
[UNI]	Unity, [Online]. Available at: https://en.wikipedia.org/wiki/Unity_(game_engine) [Accessed: 24 February 2023].

[UWS]	Unity Website, [Online]. Available at: https://unity.com [Accessed: 24 February 2023].
[VBM+21]	D. de Vleeschauwer et al., "5Growth Data-Driven AI-Based Scaling," 2021 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit), Porto, Portugal, 2021, pp. 383-388, doi: 10.1109/EuCNC/6GSummit51104.2021.9482476.
[VIC]	log4shell-kubernetes [Online]. Available at: https://github.com/vicenteherrera/log4shell-kubernetes [Accessed: 8 March 2023].
[VIR+16]	A. Virdis, G. Stea, and G. Nardini, ‘‘Simulating LTE/LTE-advanced networks with SimuLTE,’’ in Advances in Intelligent Systems and Computing, vol. 402. Cham, Switzerland: Springer, Jan. 2016, pp. 83–105, doi: 10.1007/978-3-319-26470-7_5.
[VPP]	VPP, [Online]. Available at: https://wiki.fd.io/view/VPP [Accessed 24: February 2023].
[VSS+10]	Vasan, A., Sivasubramaniam, A., Shimpi, V., Sivabalan, T., & Subbiah, R. (2010, January). Worth their watts?-an empirical study of datacenter servers. In HPCA-16 2010 The Sixteenth International Symposium on High-Performance Computer Architecture (pp. 1-10). IEEE.
[Wat89]	Watkins, C.J.C.H. (1989). Learning from delayed rewards. PhD Thesis, University of Cambridge, England.
[YOG]	Kolla-ansible Yoga, [Online]. Available at: https://docs.openstack.org/kolla-ansible/yoga/user/quickstart.html [Accessed: 24 February 2023].
[YSH+19]	Yu, Y., Si, X., Hu, C., & Zhang, J. (2019). A review of recurrent neural networks: LSTM cells and network architectures. Neural computation, 31(7), 1235-1270.
[ZEE]	Zeek, [Online]. Available at: https://zeek.org [Accessed: 24 February 2023].