Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D
GOBIERNO DE ESPAÑA

SUCCESS-6G: DEVISE

WP2 Deliverable E6

System architecture design

Project Title:	SUCCESS-6G: DEVISE
Title of Deliverable:	System architecture design
Status-Version:	v1.0
Delivery Date:	12/01/2024
Contributors:	Francisco Paredes, Miguel Fornell (Idneo), Carmen Vicente, Javier Santaella (Cellnex), Charalampos Kalalas, Ricard Vilalta, Raul Muñoz, Roshan Sedar, Pavol Mulinka, Miquel Payaro, Adriano Pastore, Jesus Gomez (CTTC), Jose Cunha, Guillermo Candela (Optare) Angelos Antonopoulos, Maria Serrano (Nearby Computing)
Lead editor:	Optare
Reviewers:	Charalampos Kalalas (CTTC)
Keywords:	System architecture; design; service level requirements; facilities

Document revision history

Version	Date	Description of change
v0.1	05/10/23	Initial proposal for the table of contents
v0.2	26/10/23	Modified table of contents
v0.3	15/11/23	Main content added
v0.4	29/11/23	Additional contents added. Version discussed during F2F meeting
v0.5	13/12/23	Final inputs
v0.6	27/12/23	Overall revision
v1.0	12/01/24	Revision and final version generated

Disclaimer

Acknowledgment

The research conducted by SUCCESS-6G - TSI-063000-2021-39/40/41 receives funding from the Ministerio de Asuntos Económicos y Transformación Digital and the European Union-NextGenerationEU under the framework of the “Plan de Recuperación, Transformación y Resiliencia” and the “Mecanismo de Recuperación y Resiliencia”.

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

Executive Summary

The key research objectives underpinning SUCCESS-6G reside in the design of a robust, secure, and computationally efficient framework that builds on the extracted knowledge from vehicular streams to offer: i) *real-time vehicle condition monitoring and fault provisioning*, and ii) *over-the-air vehicular software updates in an autonomous manner*.

This deliverable establishes the technical architecture on top of which the SUCCESS-6G solutions will be developed. The deliverable not only includes the description of the architecture for each use case in the scope of the SUCCESS-6G-DEVISE project but also provides a deep insight into the different modules and services that conform to the architecture. In addition, for each use case, the document offers the workflow definitions, providing a step-by-step description of the scenarios targeted by SUCCESS-6G-DEVISE. Each workflow describes the overall scenario and services to be supported, the involved actors and their roles, and the flow of events that form the services. There is also a section deepening the information on the different interfaces involved in each step. Finally, an initial set of functional and non-functional requirements is defined for each architecture module.

The deliverable is structured in the following sections:

- Introduction
- General system architecture for the SUCCESS-6G project
    - System architecture-specific realizations for Use Cases 1 and 2
- SUCCESS-6G modules description
- Workflow description for each Use Case and User Story
- Functional and non-functional system requirements
- Interface definition
- Conclusions

Table of Contents

List of Figures

List of Tables

1 Introduction

Modern vehicles are progressively transforming into sophisticated computing units able to gather, process, and exchange information with each other and with relevant entities. Equipped with on-board units (OBUs), vehicles can perform sensor data interactions with neighboring vehicles, roadside units (RSUs) and cloud applications, over wireless connectivity. SUCCESS-6G will bring several novelties for emerging vehicular services in 5G-enabled networks.

On previous deliverables, the SUCCESS-6G use cases (***vehicular condition monitoring and fault provisioning*** and ***automated software updates for vehicles)*** were described, including, for each use case, three user stories, corresponding to each of the subprojects (EXTEND, DEVISE, VERIFY) of SUCCESS-6G. Each user story described the overall scenario and service to be supported, the involved actors and their roles, the flow of events that form the service, relevant pre-conditions, requirements/constraints, and other information.

Going further in the development on the SUCCESS-6G project we propose, in this document, the general architecture that will support the functionalities required to accomplish the targets related to the SUCCESS-6G-DEVISE use cases, using the innovations that were initiated in the E5 deliverable. Parting from the standard ETSI TS 123 287, which has the purpose of standardizing the "Application Layer (AP) for V2X (Vehicle-to-Everything) communication systems", we will provide the related enhancements in the sections 5 to 8 to improve the 5G functionalities with the aid of AI/ML modules to support the main goals of the SUCCESS-6G-DEVISE project. The proposed architecture will be the base for the technical contributions to be developed within WP3-5, and will be validated within the aforementioned WPs, as well as through SUCCESS-6G-DEVISE proof-of-concept demonstrations to be specified within WP4-5.

2 General Architecture

Figure 1 illustrates the common general architecture that will be used as a framework for the different vehicular use cases. This is the common background for the project and will be individualized for the two SUCCESS-6G use cases, since each use case entails different modules that will be specified in the corresponding sections. We provide the details in the following.

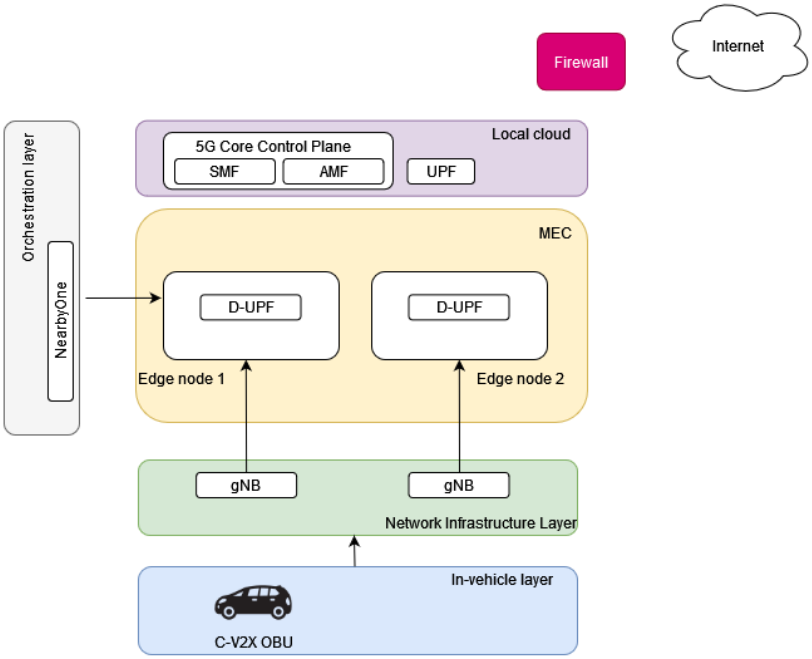

Figure 1 Generic system architecture supporting the different vehicular use cases in SUCCESS-6G

2.1 Specific system architecture for vehicular condition monitoring

Figure 2 illustrates the system architecture for vehicular condition monitoring and predictive maintenance (Use Case 1). The different modules/layers comprising the system architecture for this use case are described as follows:

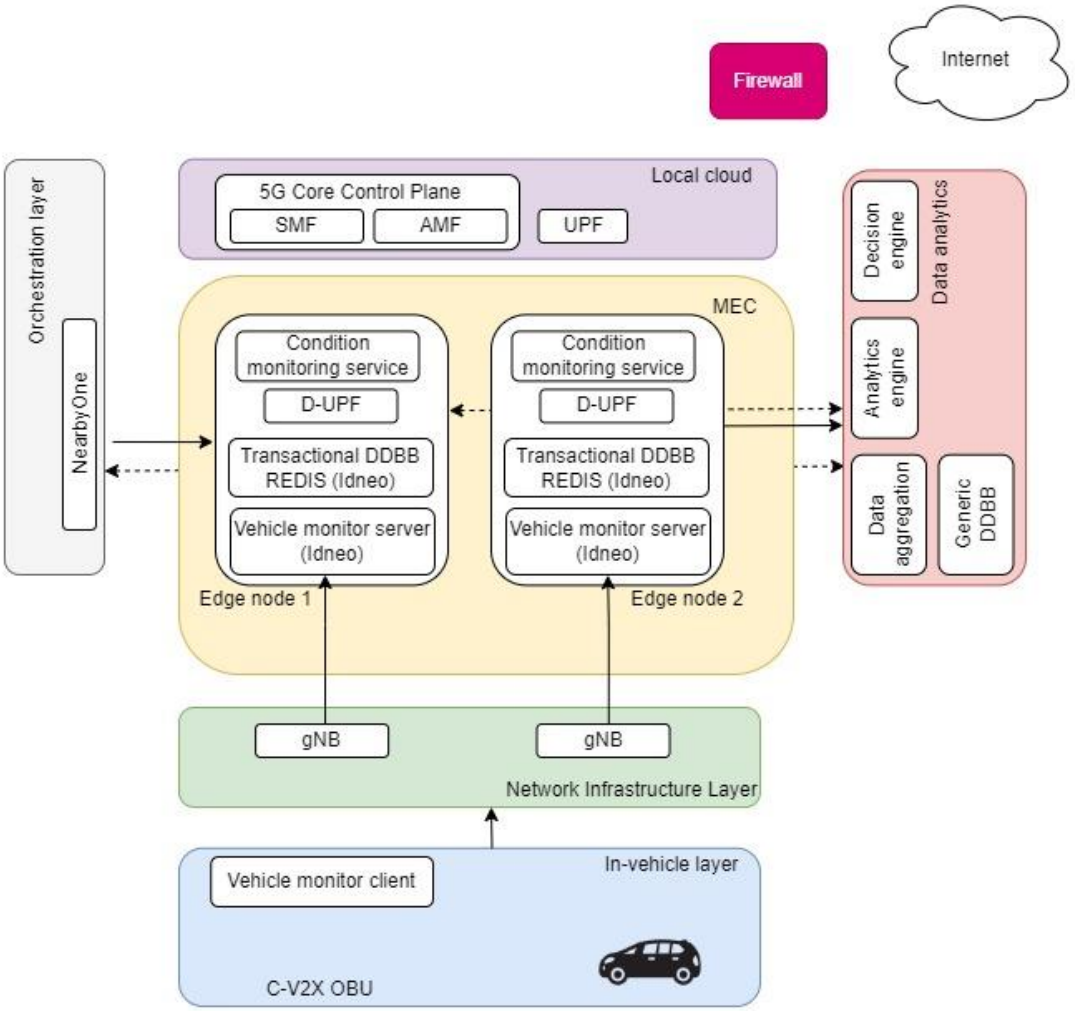

Figure 2 System architecture for vehicular condition monitoring

In-vehicle Modules: The vehicle's architecture involves an On-Board Unit (OBU) connected to the OBDII bus, extracting data from various car sensors. The OBU is equipped with a precise Global Navigation Satellite System (GNSS) module, achieving accuracies of 5cm with a real-time kinematic (RTK) positioning GNSS station. There are two antenna modules: a patch-type GNSS antenna installed centrally on the roof for satellite positioning and 4 MIMO 5G antennas providing 5G connectivity, initially placed on the trunk cover. The OBU is powered by the accessory battery rail.

Vehicle data is sent to an edge server for decoding, formatting, and storage in a CAR database. Communication between the vehicle and the gNB occurs via 5G SA or a private network, and a firmware-over-the-air (FOTA) server manages OBU software updates. Optionally, the vehicle can receive RTK corrections from a nearby base station for enhanced position accuracy.

Network Infrastructure Layer: Network infrastructure supports a complete non-roaming 5G Private Network in the Castelloli Mobility Lab, composed by 5G-SA Core, MEC and 5G-New Radios for C-V2X and other novel intelligent transport system (ITS) services in the mobility scope.

Condition Monitoring Service: It is responsible (with the aid of Data Analytics layer) for the following activities within the Multi-Access Edge Computing (MEC) infrastructure related to the monitoring of the condition of the vehicle and the identification of abnormal events:

- Data preprocessing - analysis, selection of features and reduction of feature space.
- Outlier detection and interpretation - Specific methods and models for outlier detection and interpretation.
- Classification of events - Evaluation and tuning of classification algorithm for best results.

Orchestration Layer: The orchestrator layer is responsible for the service onboarding and lifecycle management (LCM) of cloud-native applications and infrastructure at a global scale, across the edge and cloud sites. The NearbyOne Orchestrator plays the role of the orchestration engine in Use Case 1 of SUCCESS-6G.

Local Cloud: SUCCESS-6G local cloud provides a system of virtual machines that support different services of the 5G network, besides MEC applications for the end2end. Local cloud is supported by the ICT infrastructure located in the Control Room of Castelloli Mobility Lab. It is composed basically of virtualized machines running on a server.

Data Analytics Layer: The Data Analytics layer includes a wide variety of responsibilities related to the Data Analytics/Machine learning procedures. On one side, it is responsible for exploratory data analysis and its corresponding visualizations. This task will be implemented as part of NearbyOne Observability Stack, using Grafana dashboards and reports, Prometheus to export system metrics and Kepler and Thanos to export and keep the information. On the other side, we also locate in this module the MLOps structure that will allow us to obtain the model that will infer and predict the information obtained from the vehicle sensors. The different tools and models suggested for this task are expanded in the corresponding section.

2.2 Specific system architecture for over-the-air vehicular software updates

The illustration in Figure 3 intricately outlines the proposed system architecture, meticulously aligning with the overarching SUCCESS-6G framework and tailored to cater to the intricacies of Use Case 2, specifically focusing on over-the-air software updates. The assortment of components showcased in the diagram seamlessly collaborates to establish a resilient Vehicle-to-Everything (V2X) connectivity framework, ensuring steadfast security measures, and optimizing computational resources with a heightened emphasis on efficiency.

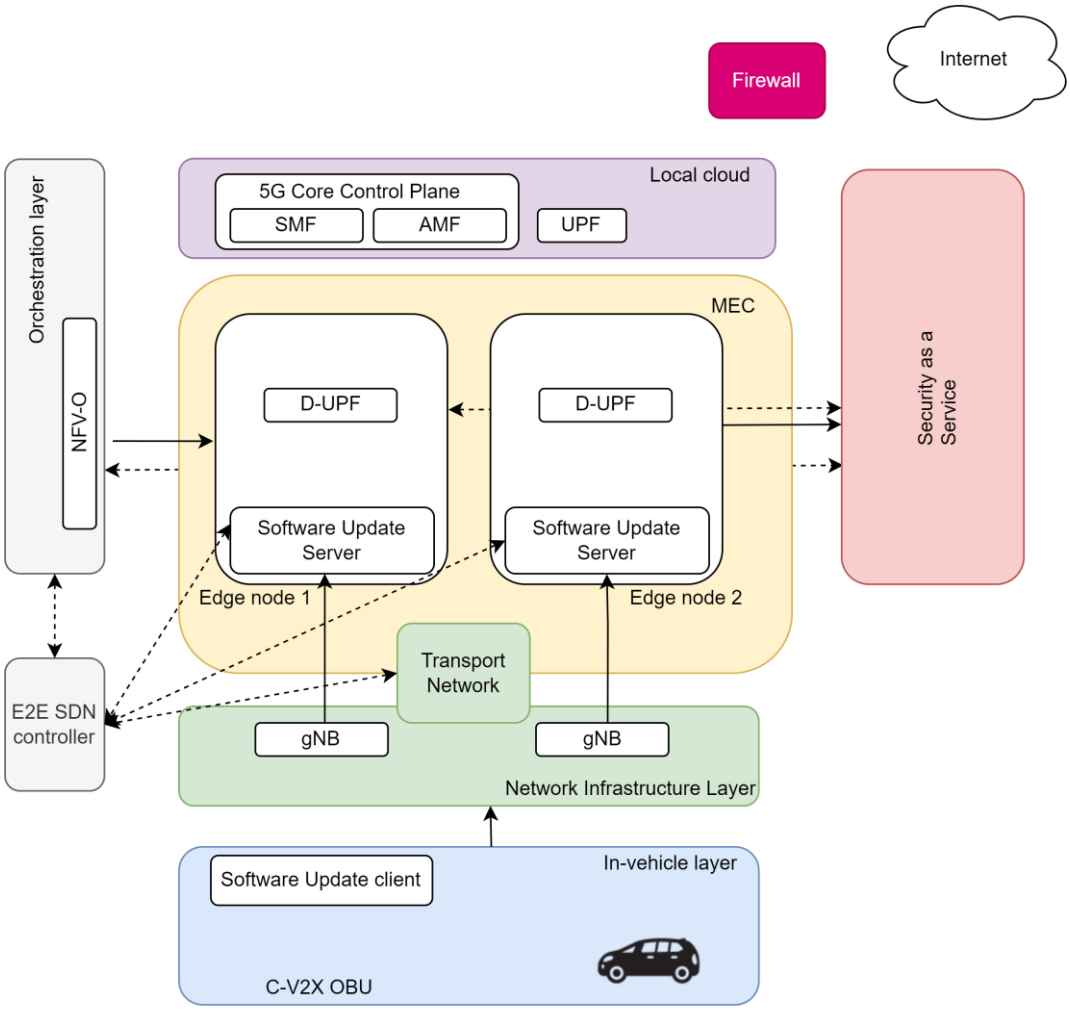

Figure 3 System architecture for over-the-air software updates

At the core of this architectural marvel lies the Security as a Service component, entrusted with the responsibility of furnishing a secure conduit for over-the-air software updates between the server, housed within the MEC infrastructure, and the client applications. To fortify this security infrastructure, probes are strategically deployed atop the Transport Network, serving as vigilant watchdogs that scrutinize the traffic flow between client-server applications. These probes are designed to detect any aberrant or unexpected behaviors, acting as a preemptive defense mechanism against potential threats.

Ensuring the robustness of the traffic flow connectivity is a multifaceted endeavor, integrating the inherent location-awareness of the numerous edge nodes into the End-to-End (E2E) Software Defined Networking (SDN) controller. The amalgamation of Global Positioning System (GPS) coordinates

augments the informational spectrum of both network endpoints and Connectivity Service Requests. To optimize this spatial information, a sophisticated algorithm has been ingeniously crafted within the E2E SDN controller. This algorithm, a product of meticulous design, strives to match the optimal location that best aligns with the incoming requests and the designated endpoints, thereby fortifying the connectivity and enhancing the overall reliability of the system.

In the pursuit of furnishing efficient computational resources, the interface between the client application and the MEC server plays a pivotal role. This is achieved through the integration of a novel European Telecommunications Standards Institute (ETSI) MEC 015 API, seamlessly embedded within the E2E SDN controller. The primary objective of this API is to usher in a new era of bandwidth management support at the Transport Network level, ensuring that computational resources are allocated judiciously, attuned to the specific needs and bandwidth requirements of the applications in play.

In essence, this comprehensive system architecture not only champions the deployment of over-the-air software updates but also establishes a resilient V2X connectivity framework, fortifies security through proactive monitoring, leverages spatial intelligence for optimal connectivity, and judiciously allocates computational resources, ushering in a paradigm shift in the landscape of vehicular connected systems.

3 In-vehicle Layer Description

3.1 Specific implementation for vehicular condition monitoring

3.1.1 In-vehicle architecture for vehicular condition monitoring with robust V2X connectivity

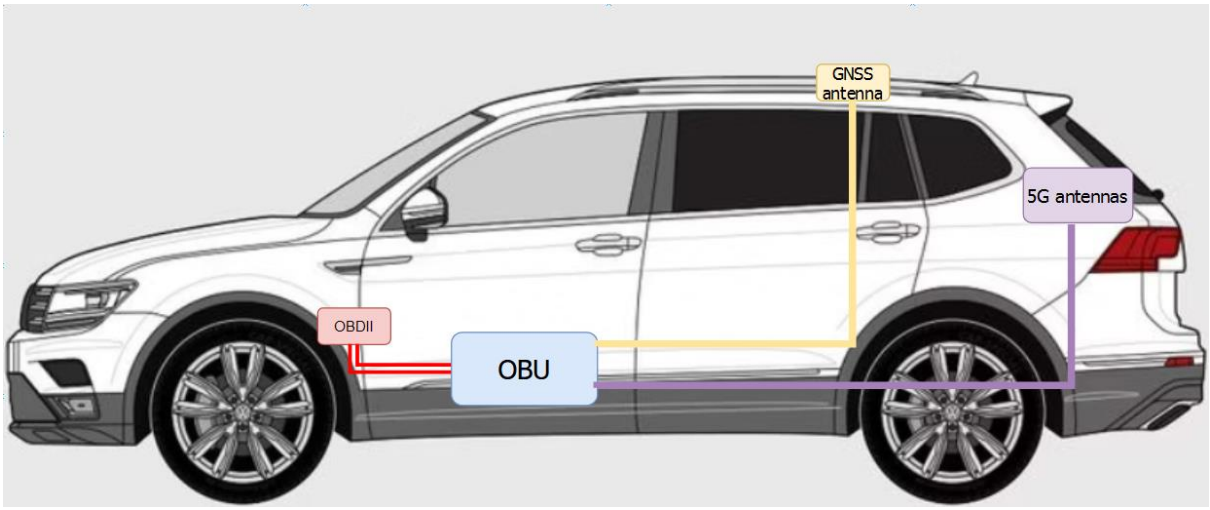

Figure 4 On Board Unit (OBU) illustration

As illustrated in Figure 4, the in-vehicle architecture mainly consists of an OBU connected to the OBDII bus, which allows us to extract information from both the public and private car sensors. To enhance the real-time sensor data with more value, the OBU unit has been equipped with a highly precise GNSS module, enabling us to achieve accuracies of up to 5cm with the assistance of an RTK GNSS station.

Regarding the antennas, the vehicle will be equipped with two antenna modules. The first module is the GNSS antenna. This antenna is of the patch type and should be installed in the center of the vehicle's roof, ensuring at least a 50cm radius of metallic surface beneath it. The installation will not be permanent and can be placed magnetically. The second module of antennas is responsible for providing 5G connectivity to the OBU. Within this module, 4 MIMO 5G antennas will be integrated, covering the entire NR 5G frequency range. The installation will not be permanent and, initially, will be installed in the trunk cover.

Regarding the power supply for the devices, only the OBU will need to be powered by the accessory battery rail. For the SUCCESS-6G project, a Volkswagen Tiguan or similar vehicle will be used.

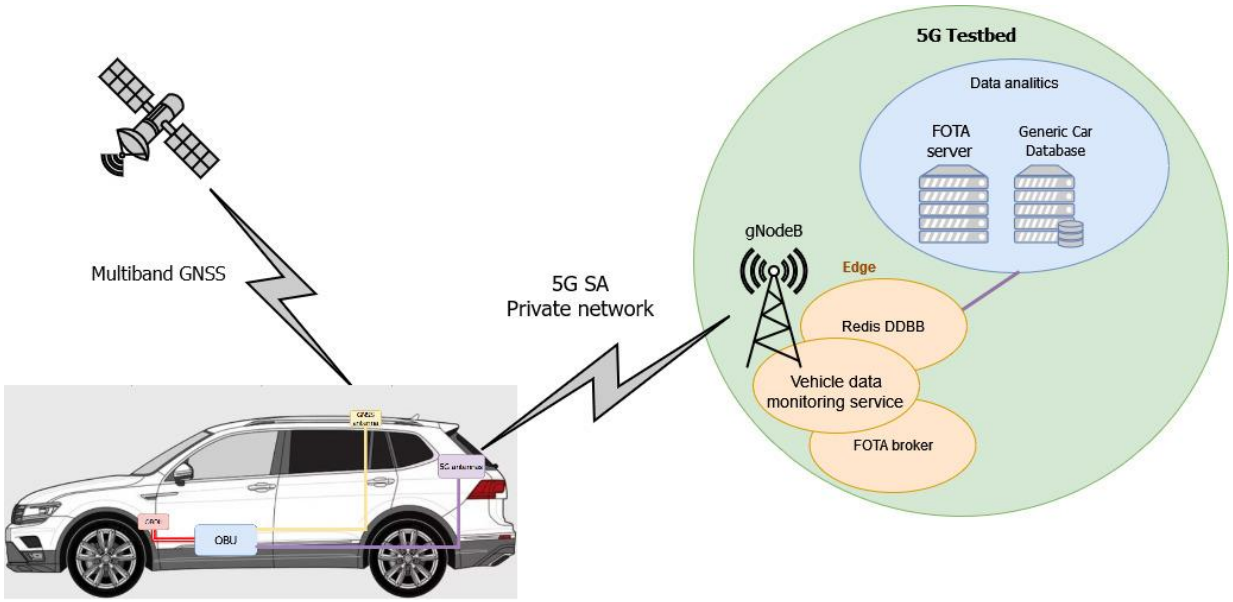

Figure 5 General on-site architecture for vehicular condition monitoring with robust V2X connectivity

As illustrated in Figure 5, the system architecture consists of the vehicle, which, with the aid of an OBU, will be able to read real-time sensor information from the vehicle's internal bus. These vehicle data will be sent to the edge in real-time, where an edge server will decode them, format them, and store them in the CAR database, as shown in Figure 6. The communication between the vehicle and the gNB will be carried out using 5G SA, and optionally, on a private network. Additionally, a FOTA server will be required, which will be responsible for managing software updates for the OBU.

To complete the architecture, optionally, the vehicle can receive RTK corrections from a nearby base station to enhance the equipment's position accuracy up to 5cm.

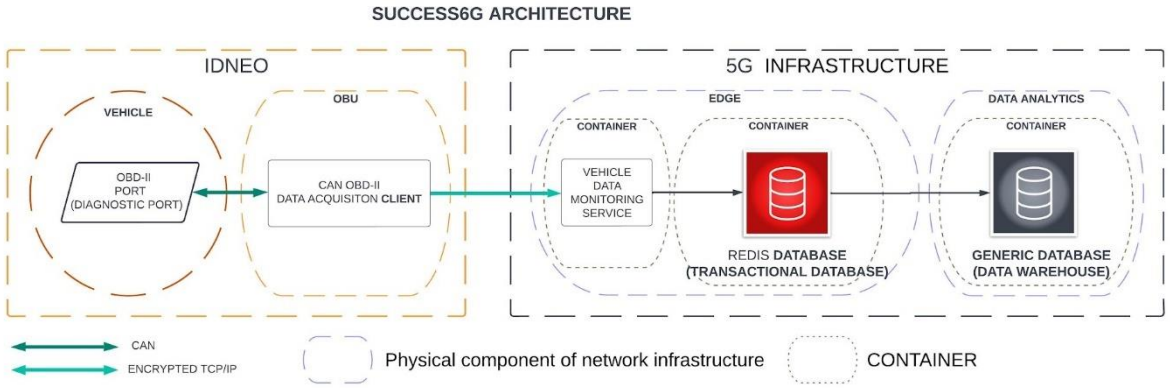

Figure 6 General architecture with edge component for vehicular condition monitoring with robust V2X connectivity

3.1.2 Vehicular condition monitoring with security guarantees

In order to ensure security during vehicle status monitoring, an architecture based on TLS 1.3 for the upload session will be employed, as illustrated in Figure 7.

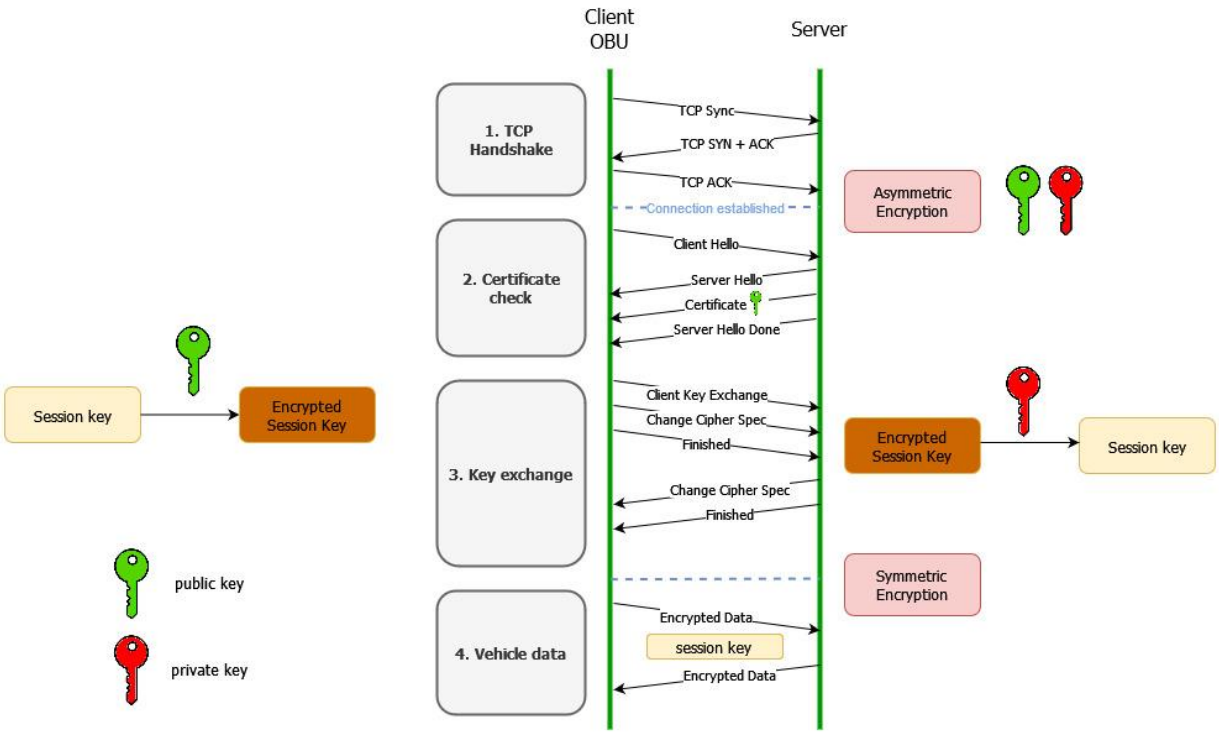

Figure 7 Sequence diagram for vehicular condition monitoring with security guarantees

TLS 1.3 is a cutting-edge cryptographic protocol designed to secure online communications. It marks a significant advancement by prioritizing modern encryption techniques, shedding outdated algorithms, and simplifying the handshake process. This streamlined approach results in quicker connection setup times and significantly enhances security. TLS 1.3 emphasizes Perfect Forward Secrecy by default, ensuring that even if encryption keys are compromised, previous and future sessions remain protected. Its fortified resistance against cyber threats, like downgrade attacks, and support for Zero Round-Trip Time (0-RTT) mode, catering to faster connections for returning users, establishes TLS 1.3 as a robust safeguard for ensuring data privacy and integrity over networks.

3.2 Specific implementation for over-the-air vehicular software updates with robust V2X connectivity

In automotive devices, firmware updates are typically deployed during the shutdown procedure. For this purpose, the device includes a state machine that manages a controlled shutdown.

Firmware updates generally fall into two primary categories: the first one updates the entire operating system, while the second focuses on minor changes, such as specific applications or security patches, within the operating system. In the context of SUCCESS-6G-DEVISE project, the focus lies on executing complete overhauls of the operating system while incorporating policies that will govern the download process.

Use Case 2 considers two paradigms of FOTA updates, single OBU and multi-ECU FOTA using the same architecture based on open-source resources.

3.2.1 Single OBU FOTA

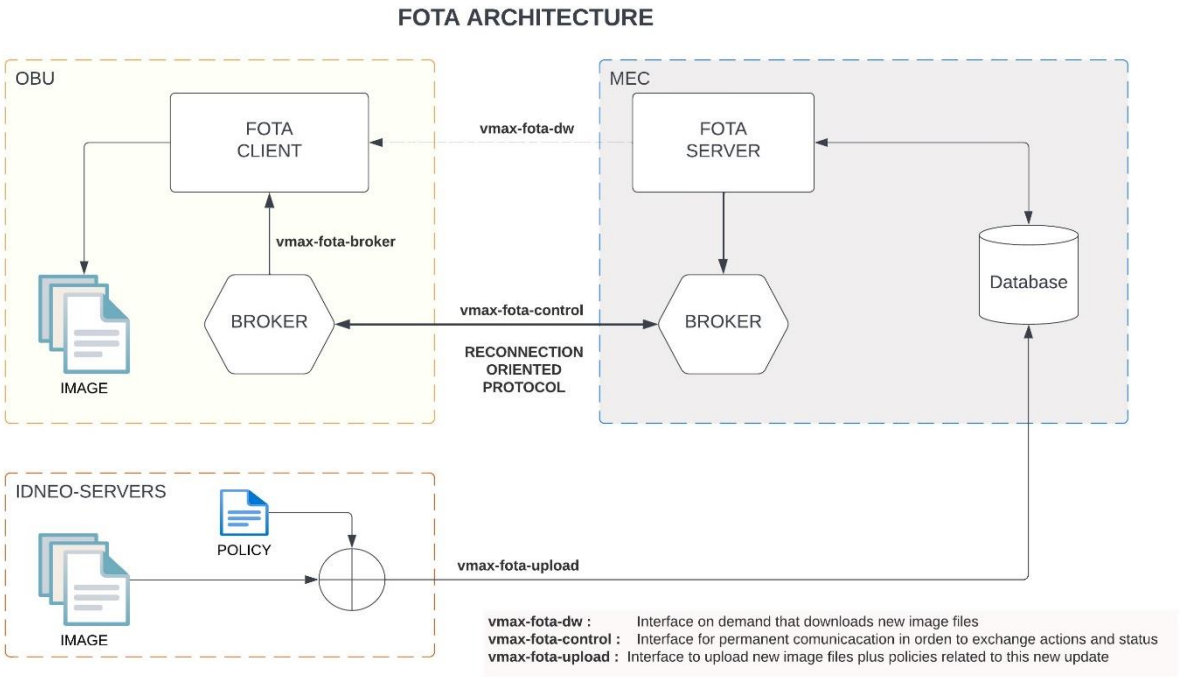

Figure 8 Functional blocks for over-the-air vehicular software updates with robust V2X connectivity. Single OBU FOTA

Figure 8 describes the functional blocks for over-the-air vehicular software updates (Use Case 2) with robust V2X connectivity for single OBU FOTA. The procedure of software update is detailed as follows:

1. New software release (OBU's image) is generated by IDNEO.
2. The release is uploaded with policies to a FOTA SERVER in MEC.
3. FOTA SERVER monitors the permanent connection with OBU and launches a new release notification based on the notification policies, using vmax-fota-control interface.
4. FOTA CLIENT receives the notification, also download policies.
5. FOTA CLIENT proceeds to download the software release based on the download policies, using vmax-fota-dw interface.

6. New software release is deployed by OBU, by internal power management procedure.

3.2.2 Multi-ECU FOTA

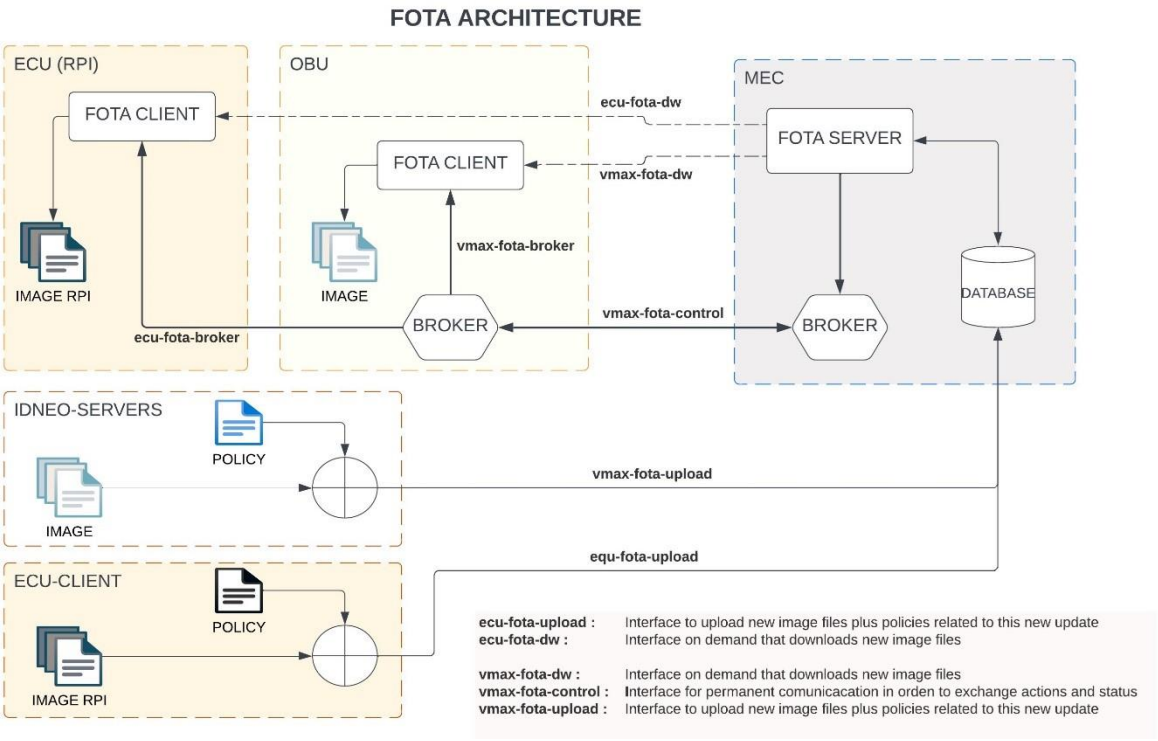

Figure 9 Functional blocks for over-the-air vehicular software updates with robust V2X connectivity. Multi-ECU FOTA

Figure 9 describes the functional blocks for over-the-air vehicular software updates with robust V2X connectivity (Use Case 2) for multi-ECU FOTA. The procedure of software update is detailed as follows:

1. New external ECU software release is generated.
2. The release is uploaded with policies to a FOTA SERVER in MEC.
3. FOTA SERVER monitors the permanent connection with external ECU and launches a new release notification based on the notification policies.
4. FOTA CLIENT receives the notification, also download policies.
5. FOTA CLIENT proceeds to download the software release based on the download policies, using vmax-fota-dw interface, provided by OBU VMAX using the Uu channel.
6. New software release is deployed by the external ECU, by its internal power manager procedure.

4 Network Infrastructure Layer Description

Network infrastructure supports a complete non-roaming 5G Private Network in the Castelloli Mobility Lab (Figure 10), composed by 5G-SA Core, MEC and 5G-New Radios for C-V2X and other novel ITS services in the mobility scope.

The 5G Private Network is composed by:

- 1 5G-Core SA, service-based.
- 1 Central UPF
- 2 Distributed UPF
- 2 MEC
- 2 5G-NR (BBU + RRU) 100 MHz in band n77. (3800 – 3900 MHz)

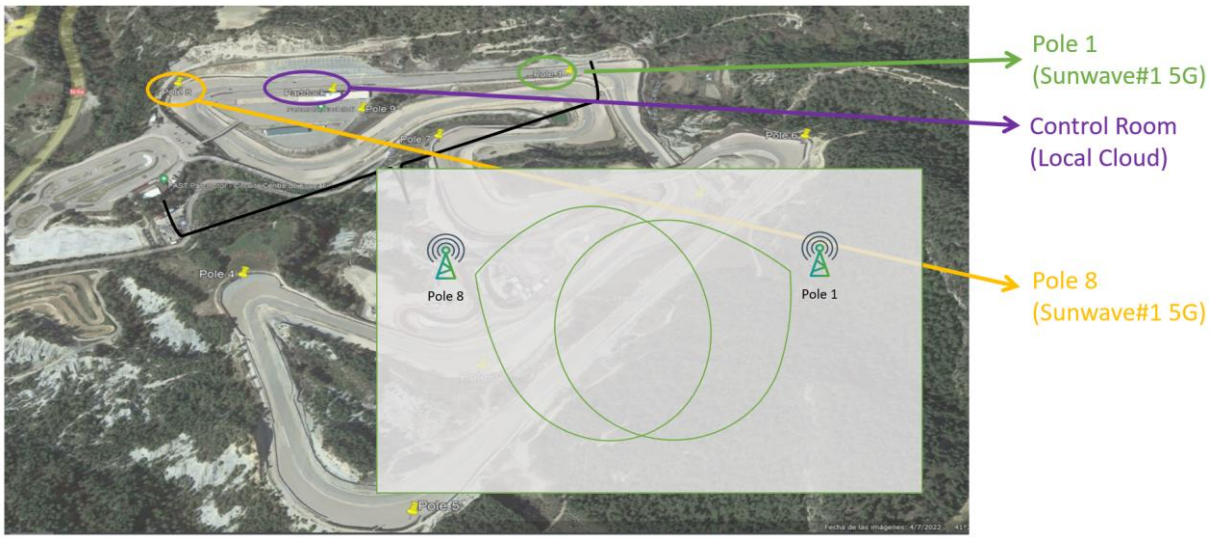

Figure 10 Location of gNBs and radio sectors in Castellolí test site

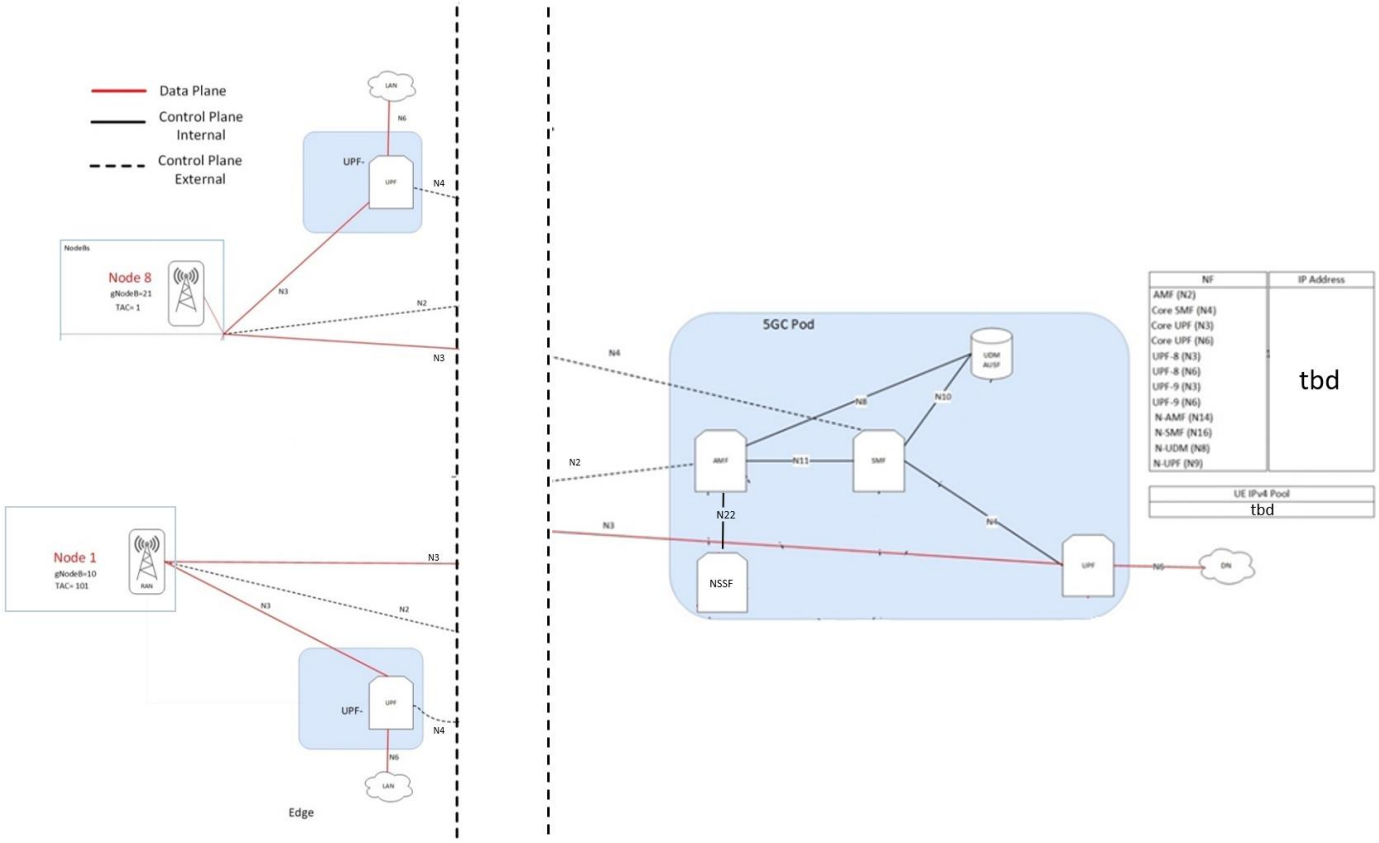

Figure 11 Internal architecture of the 5G SA networks deployed in the small-scale test site of Castellolí

4.1 5G-SA Core Network

As shown in Figure 11, the 5G Core deployed in Castellolí has been supplied by Druid, an Irish company specialized in 4G/5G private network solutions. The solution is composed of a 5G Core instance, running on different servers. The deployed solution consists of a 5G Stand-Alone (SA) configuration with centralized and distributed UPFs. There are two distributed UPFs, one associated to the node 1 and the other to the node 8 as shown in Figure 11. Druid provides a technology platform called Raemis, which consists of cellular software assets optimized for business use cases. This platform offers mobile communication solutions that are tailored to meet the needs of enterprises.

The Raemis functionality that has been implemented has the following characteristics:

- Dashboard (Figure 12)
- 5G private network with private subscribers, private cell network, N2 handover, Idle mode cell reselection, UE attachment/implicit detach/re-attach, and data service.
- Real-time System Monitoring.
- Cells Management.
- Network Management including 5G data slicing.
- Alarm Monitoring.
- System management.
- Traffic separation.

- QoS of Service per user group.
- Access control per user or user group.

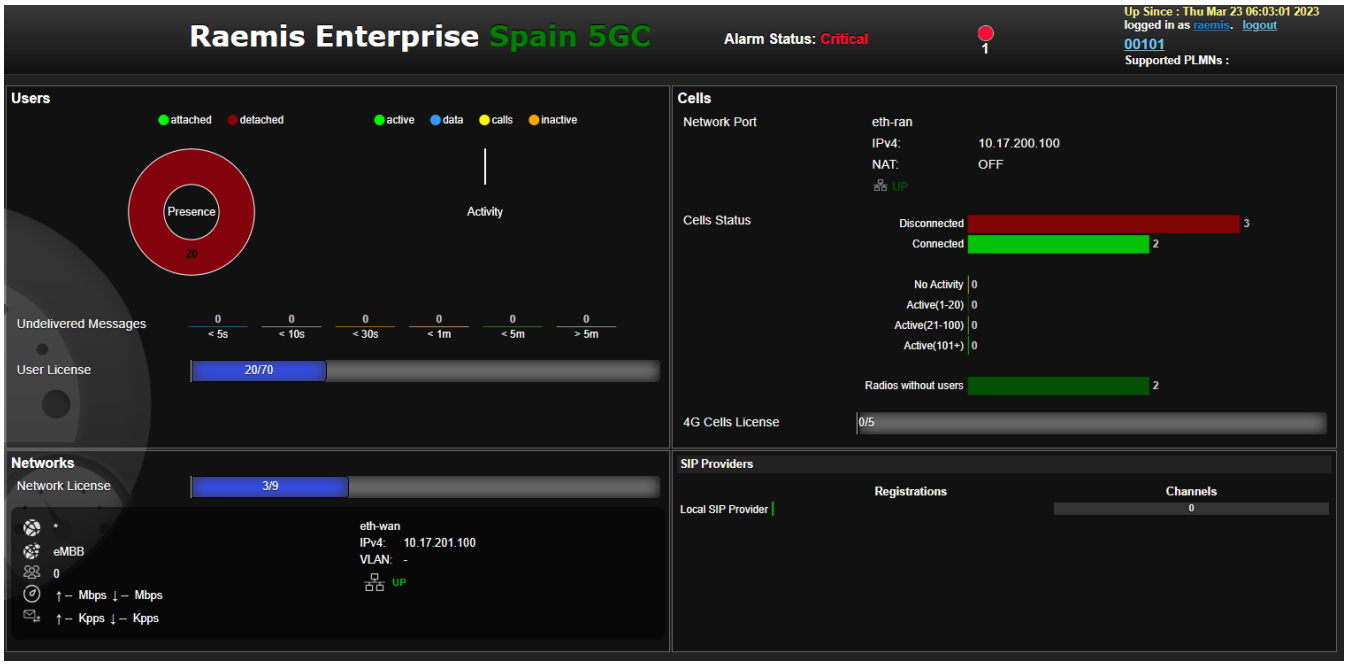

Figure 12 *Raemis dashboard*

This implementation matches with the design of the 5G SA network that is targeted for the deployment shown in Figure 2 and Figure 3. One 5G Core deployed in two separated Lenovo SR-650 servers (virtualized) placed in the same rack as depicted in Figure 13.

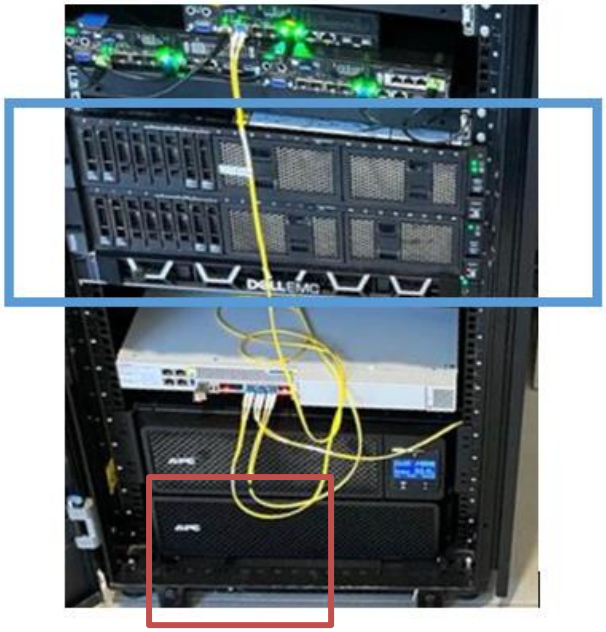

Figure 13 Castellolí Rack with SR650 Lenovo servers that host the 5G Core

In order to deploy the 5G SA Core in the virtualized SR650, both servers have installed vSphere (vmware), a software that allows to create different virtual machines and allows to virtualize the resources (Figure 14 and Figure 15).

Figure 14 Vmware vSphere platform

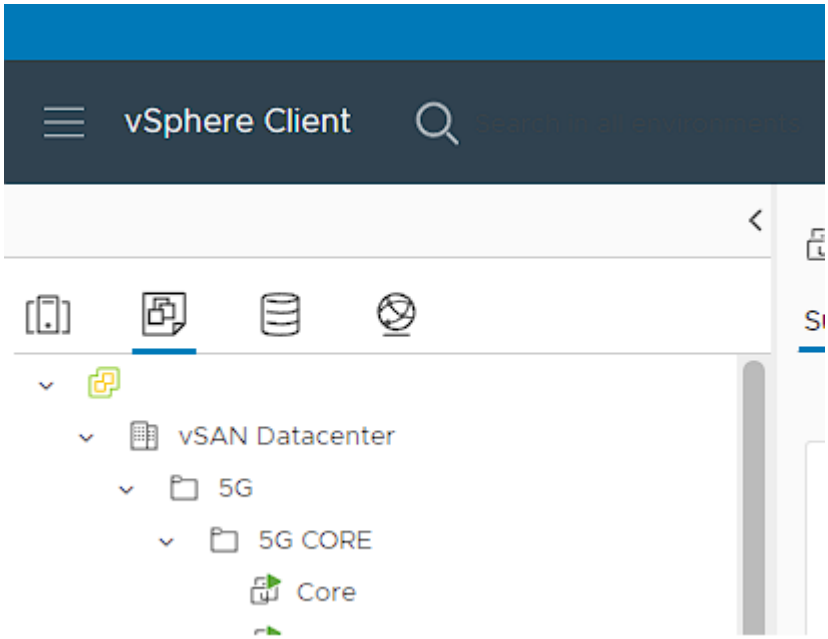

Figure 15 vSphere Client where VMs are allocated

The characteristics of the virtual machine for the 5G SA Core network deployment are as follows:

- CPU: 16 CPU(s)
- Memory: 32 GB
- Hard Disk: 100 GB
- OS: CentOS 8 (64-bit)

- VLAN: tbd

To get access to this VM and the other equipment, a VPN (Virtual Private Network) will be granted allowing access to the different VLANs. In addition, Druid has provided 20 SIM cards for the PLMN-id, as shown in Table 1:

ICCID	MSISDN	IMSI
tbd	tbd	tbd

Table 1 5G SA SIM cards

4.2 MEC (Distributed UPFs)

The MEC layer deployed in the Castellolí small-scale test site consists of four SE350 Lenovo edge servers depicted in Figure 16. The small form factor of Lenovo SE350 allows to have edge computing capabilities co-located with a radio site along the corridor, thus making an efficient use of space and energy, while offering good computing capabilities as highlighted by its technical specifications detailed in Table 2.

Figure 16 SE350 Rack in Castellolí

Two distributed UPF will be allocated in two SE350, identified as SE350#1 and SE350#2 in Figure 17. Each distributed UPF will remain to one of the two 5G Sunwave Radios.

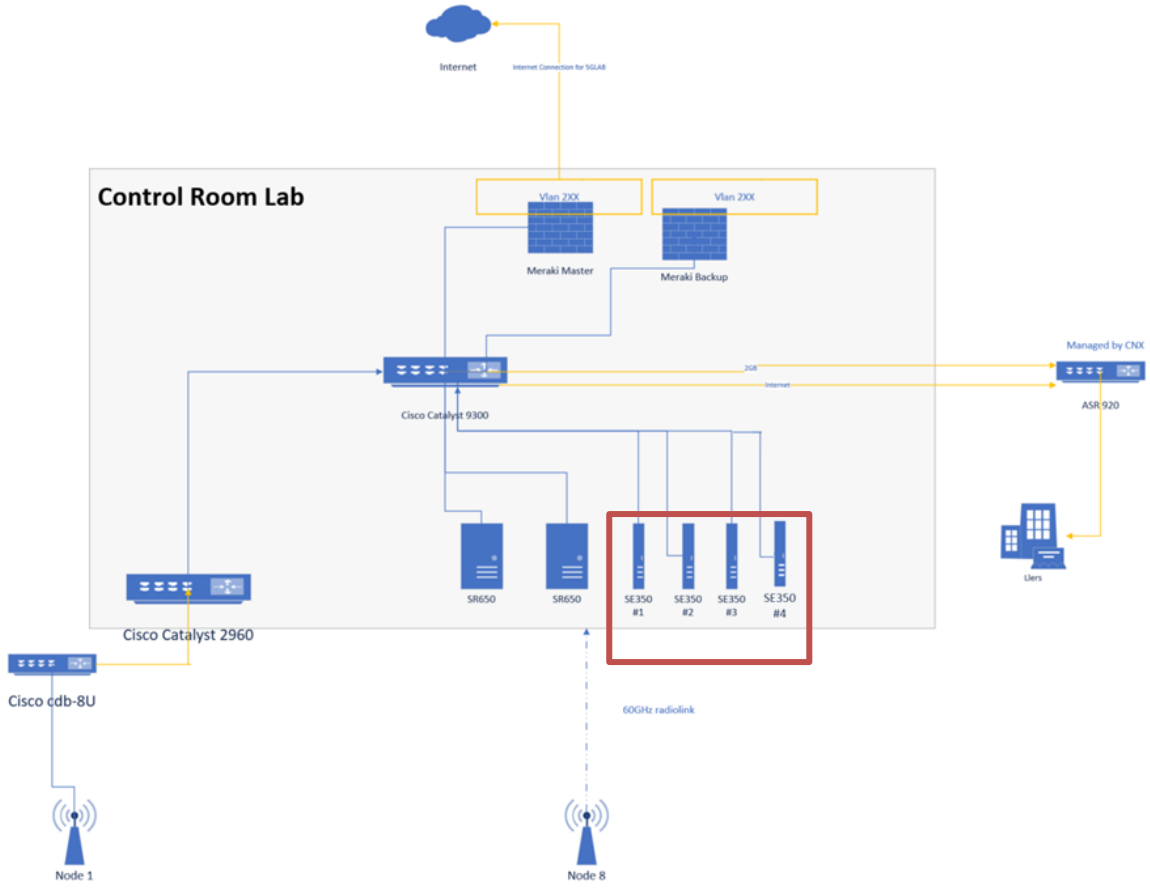

Figure 17 Network of the Castelloli architecture

The characteristics of the SE350 are as follows:

Size	43.2mm Height/ 209mm Width/ 376mm Depth
Weight	3.6 kg
Processor	One Intel® Xeon® processor D-2100 product family
Memory	256 GB (4 x 64GB LRDIMM)
Connectivity options	Ports: Two USB 3.1 / Two 1Gb Ethernet/ Two 10Gb SFP+
	WLAN: IEEE 802.11 a/b/g/n/ac
	LTE (3GPP R11)

Table 2 Lenovo SE350 technical specifications

For the SUCCESS-6G project needs, two of the four Lenovo SE350 servers host the following network functions:

- Server 1 associated to Edge Node 1: Distributed UPF node 1
- Server 2 associated to Edge Node 1
- Server 3 associated to Edge Node 8: Distributed UPF node 8
- Server 4 associated to Edge Node 8

4.3 5G-New Radios

The outdoor coverage of the 5G network is supported on SUNWAVE technology, using n77 band. It is a distributed solution composed of BBU and RRU units.

- BBU nCELL-T5000, is the 5G-NR base station unit (Figure 18). It provides central control and management of the entire base station system providing network access functions, direct access and data interaction with 5G-Core and baseband processing functions. Facilitates nGAP, XnAP interface, 5G-NR access network protocol stack functions, RRC, PDCP, SDAP, RLC, MAC and PHY protocol layer functions.
- RRU RU4370 converts fiber optic signals into cellular air technologies, to provide outdoor coverage of the 5G network.

Figure 18 5G-NR RRU module

Table 3 details the composition of each site while Table 4 details the characteristics of the gNBs at the Castellolí small-scale test site.

Site name	Cell type	MIMO type	Equipment Provider	Energy supply
Node 8	small cell	4x4	Sunwave	Self-sustainable
Node 1	small cell	4x4	Sunwave	Grid connected

Table 3 Features of the gNodeBs at the Castellolí small-scale test site

Site name	Product type	Product name	Product details	Illustration
Node 1 and Node 8 (Sunwave gNodeBs)	**Sunwave Antenna**	**JZD – 65DPG1515-3842T0**	• 4 ports antenna • 15 dBi Gain • Bandwith: 3800-4200 MHz	

Site name	Product type	Product name	Product details	Illustration
	Sunwave RRU	**sCELL-3470RRU**	• Product Name: sCELL-3470RRU • 4T4R • Max total carrier BW is 100MHz for NR • 5W (37 dBm) Output Power • Bandwith: 100 MHz in band n77. (3800 – 3900 MHz) • Weight 12Kg • 48 VDC	
	Sunwave Baseband Unit	**-**	• 2x RJ-45 100/1000 BASE-T Ethernet Port • 4 x 10Gbps ports SFP+ Ethernet Port • 4 x Radio Interface ports	

Table 4 Composition of gNBs at the Castelloli small-scale test site

5 Condition Monitoring Service Description

Vehicle Data Monitoring Service

The vehicle monitoring service is responsible for decrypting, decoding, time-tag, and processing messages received from the vehicle containing data. Concerning encryption, it will act as a server for TLS 1.3 encryption, providing the necessary certificates to clients. Regarding decoding, its role is to interpret all traffic received from the vehicle, which, due to the nature of V2X messages, is highly optimized and compressed to minimize the number of bits, thus requiring reconstruction at the destination. Additionally, it generates a timestamp for all data arriving from the vehicle's CAN bus in pseudo real-time, needing adjustment with a timestamp. Lastly, it processes the data according to the sensor identifier to align it with actual measurements, making it ready for storage and consumption by other services.

Transactional Database

The outgoing data from the Vehicle Data Monitoring Service will be stored in the transactional database. This database will facilitate storing events while maintaining data integrity in multi-access environments, ensuring very high levels of effectiveness.

Vehicle Condition Monitoring Service

It is the service responsible for:

1. Data preprocessing – analysis, selection of features and reduction of feature space
2. Outlier detection and interpretation – Specific methods and models for outlier detection and interpretation
3. Classification of events – Evaluation and tuning of classification algorithm for best results

The service consumes data from the transactional database and forwards predictions to the generic database, as shown in Figure 2. In SUCCESS-6G-DEVISE, a modular and containerized end-to-end monitoring service will be developed for the purpose of evaluation, implementation and integration of data processing and machine learning methods. For this purpose, interaction with the data analytics layer will be sought. For data prediction task, we incorporate a set of models built for the ML pipeline, as illustrated in Figure 19.

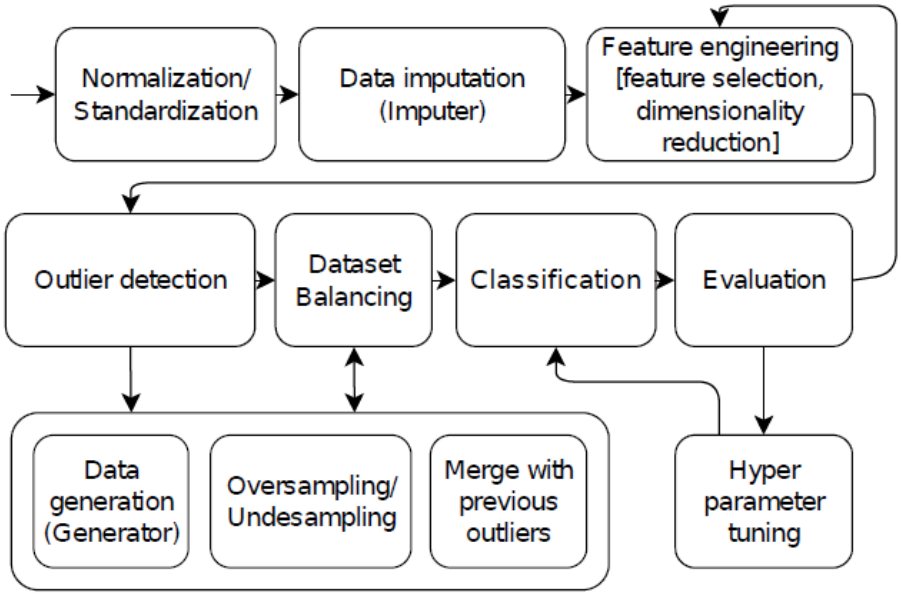

Figure 19: Machine learning pipeline used for data processing tasks, e.g., predictions of abnormal events in the vehicle [1]

The core ML pipeline used for the predictions of events consists of the following blocks [1]:

- **Normalization/standardization**: Scaling of the dataset.
- **Imputer**: Imputation of the missing data
- **Feature engineering**: Analysis, selection of features and reduction of feature space.
- **Outlier detection**: Specific methods and models for outlier detection.
- **Dataset balancing**: Oversampling/undersampling synthetic data generation methods for imbalanced dataset.
- **Classification, evaluation, tuning**: Evaluation and tuning of classification algorithm for best results.

The combination of various Python libraries, RESTful API, Kafka streams and InfluxDB time series database is chosen to keep the solution open-source, modular, scalable, and robust. All discussed components constitute pieces of publicly available open-source projects and can be easily interchangeable or removable.

6 Orchestration Layer Architecture

The orchestrator layer is responsible for the service onboarding and lifecycle management (LCM) of cloud-native applications and infrastructure at a global scale, across the edge and cloud sites. NearbyOne Orchestrator plays the role of the orchestration engine in SUCCESS-6G-DEVISE.

NearbyOne is an end-to-end cross-domain orchestration platform that is built to solve the challenges to deploy workloads at the Edge. It covers the full lifecycle management of distributed systems, networks and applications encompassing all layers, from the Edge to the Cloud, seamlessly managing all the areas of operations that take part in edge deployments: the infrastructure, the network, and the applications (as shown in Figure 20).

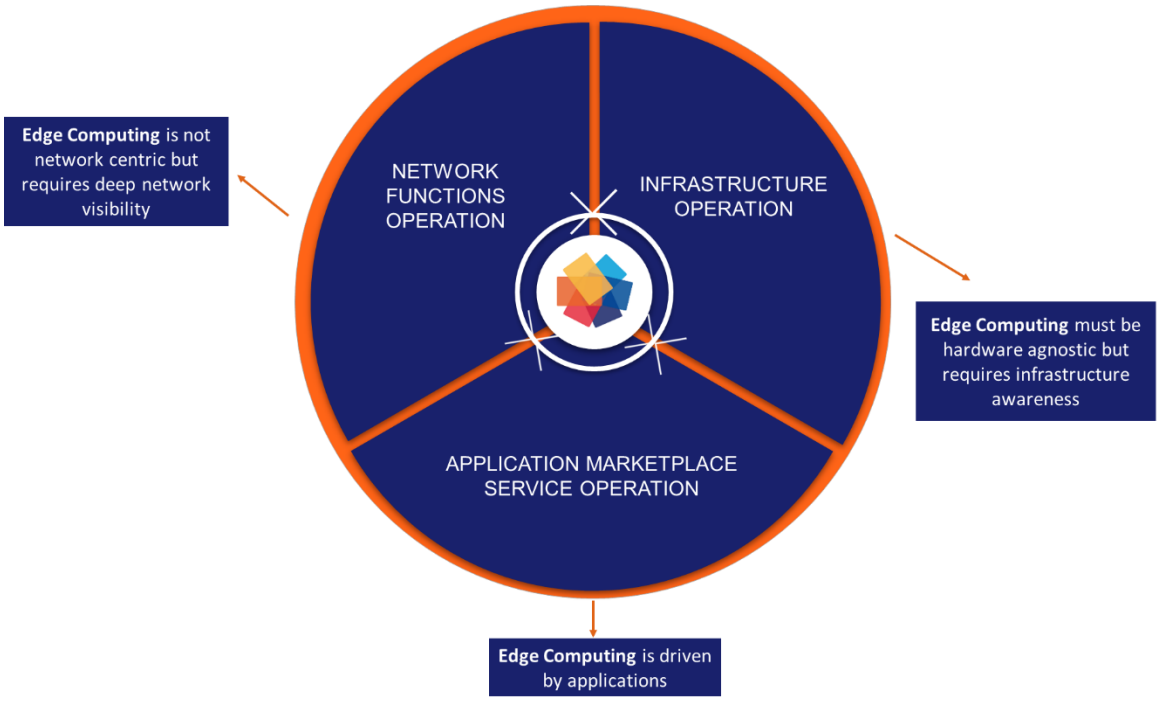

Figure 20 Fundamental pillars of any edge deployment.

- **Infrastructure**: NearbyOne allows to fully manage the lifecycle of the edge nodes as a service. It supports all kinds of infrastructure providers including Edge devices, Hyperscalers, Public, Private and Hybrid clouds.
- **xNFs,** both Virtual network functions (VNFs) and Cloud-Native Network Functions (CNFs): Onboarding of network functions that are needed to enable access to the workloads to be deployed. This includes different access technologies (ranging from IoT to 5G networks), core network components (LTE/5G cores as well as SD-WAN components), and all types of other auxiliary functions, such as firewalls or virtual routers.
- **Application and Service Operations:** Applications and services are the central component of any edge deployment, and they require the seamless integration of components provided by third-party software developers or vendors. Application developers are not required to make any changes to their application code for proper integration. The integration points between the applications and the platform (i.e., telemetry collectors or log parsers) need to be defined to provide means to NearbyOne to assess the operation of the applications and to provide

reactive actions if needed. *All configurations are made using declarative languages (YAML).*

NearbyOne is intent-driven: it allows users to onboard new applications and xNFs into the platform through the definition of a number of declarative files that define the behavior that the orchestration platform must follow to fulfil user-defined requirements.

6.1 NearbyOne architecture

The NearbyOne solution is mainly composed of:

- The *Nearby Orchestration Platform*, the main component of the solution, is in charge of performing all tasks related to the orchestration of applications and infrastructure.
- The *Nearby Blocks* are distributed components that encapsulate logic and code for different application-specific functionalities.

6.1.1 NearbyOne Orchestration Platform

Figure 21 presents the main components of the NearbyOne multi-site orchestration platform.

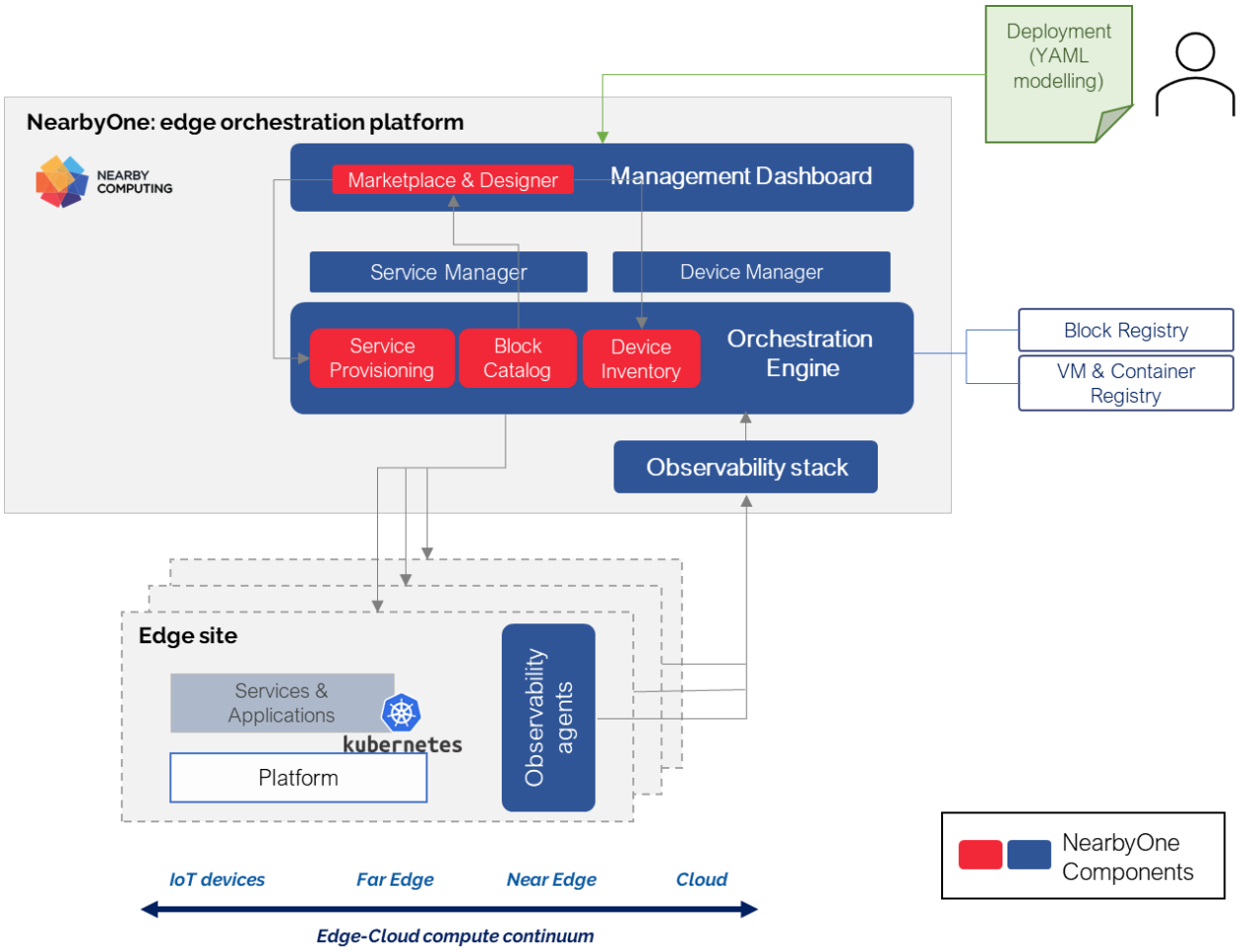

Figure 21 High level NearbyOne architecture

NearbyOne offers a full 360 view of the edge, addressing the main challenges of application, network functions, and infrastructure management through a single pane of glass and an intuitive GUI (NearbyOne ***Management Dashboard***). NearbyOne not only provides mechanisms to automate and orchestrate provisioning of services but also offers a ***Marketplace*** that facilitates the choice and chained deployment of services and applications with only one click. The *Marketplace*, easily extendible, gives access to the catalog of solutions available for rapid service deployment through the NearbyOne Dashboard (as shown in Figure 22). The ***Designer*** module of the NearbyOne Dashboard is used to define service deployment chains and establish the relations among them. Deployments, in the form of YAML manifests, contain not only the application deployment information, but also its requirements and placement/configuration policies.

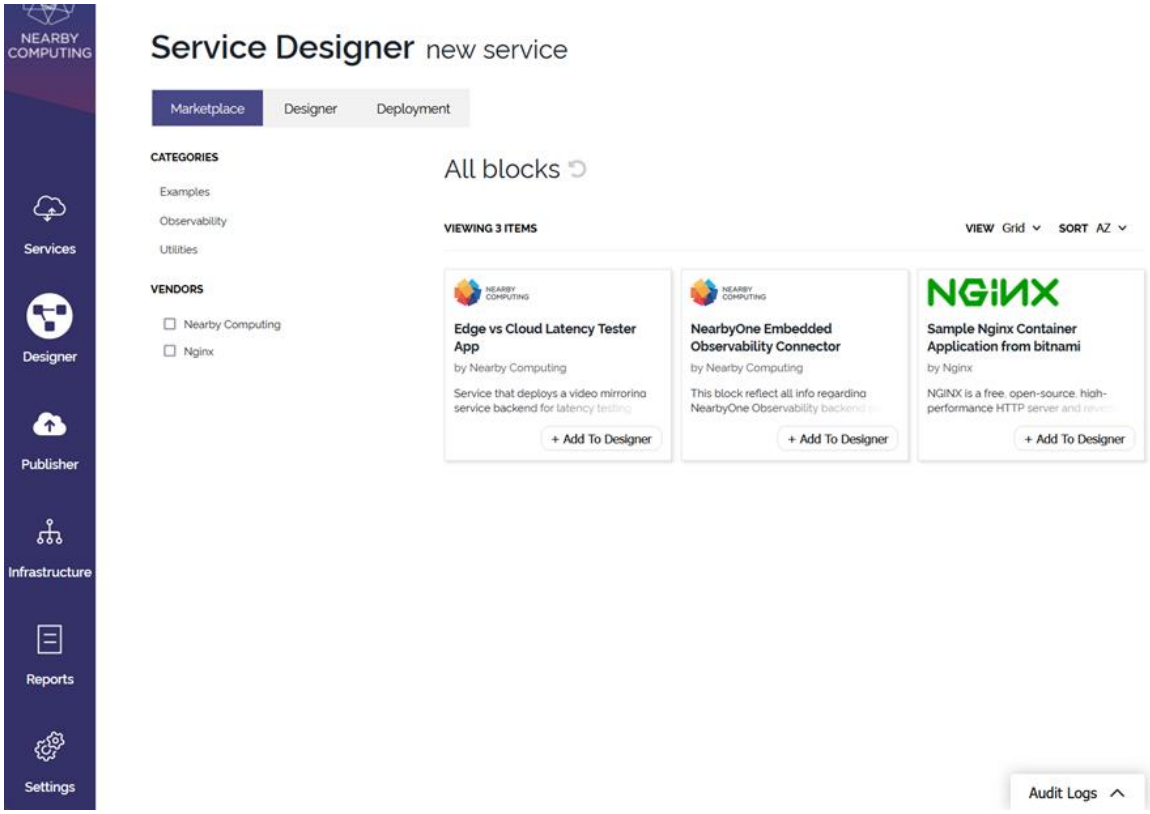

Figure 22 NearbyOne dashboard Marketplace

At the core of the orchestrator, the ***Device manager*** is responsible for the operations related to edge sites, while the ***Service manager*** is responsible for managing high-level CRUD (create, read, update, and delete) operations for service chain deployments and other components related to them. The ***Orchestration engine*** is the main operational component responsible for managing complex and distributed operations, enforcing eventual consistency rather than transactions. It follows the pattern of self-healing reconciliation loops. It also includes components that manage the policies and other dynamic behavior related to services and devices. The ***Block Catalog*** is responsible for keeping track of the service definitions and their associated images and binary file dependencies. The ***Service Provisioning*** manages service access control as well as service level definitions and policies associated to each service instance. The ***Device Inventory*** is responsible for maintaining the catalog of orchestrated devices tracking information such as MAC addresses, IP addresses, hardware capabilities, or provisioned flavor for each device, while also managing their provisioning and providing resource access control enforcement.

The NearbyOne Observability Stack is described in the section 6.2.

Besides the NearbyOne components, other relevant external entities necessary for the operation are the Block, and container registries, which are the repositories used to store and access application or services artifacts and container images.

6.1.2 Nearby Blocks

In NearbyOne, third-party software components onboarded into the platform to operate services deployment are named Nearby Blocks or simply Blocks. Each Nearby Block contains references to the application logic (service containers or virtual machines (VMs)), and they are encapsulated with a set of auxiliary components that provide the means for the application to be effectively managed. Nearby Blocks describe how to deploy applications and network functions, including different aspects such as:

- how to render their configurations
- where to place them across the registered clusters
- how many instances to deploy (statically or dynamically decided at runtime)
- how to monitor their KPIs
- how to use KPIs to manage the aspects defined above in this list

NearbyOne orchestration resources follow a similar syntax to Kubernetes manifests, and Nearby Blocks are packaged and follow a templating syntax similar to Helm charts [2].

6.2 NearbyOne Observability stack

The NearbyOne Observability stack, among other functionalities, collects metrics from the infrastructure and services. NearbyOne observability stack is powered by cloud-native open-source technologies commonly used across multiple domains to collect and aggregate telemetry, logs and traces. Concretely, the telemetry stack is the fundamental element of the NearbyOne observability stack to collect, transport and aggregate monitoring data. The consolidated telemetry is available to the orchestration engine to influence policies and configurations.

The fundamental elements of the NearbyOne telemetry stack, a component of the NearbyOne observability stack, are (as shown in Figure 23):

- Prometheus (https://prometheus.io): Main telemetry collection component, extensively used to collect telemetry from multiple services at a cluster level. The community around Prometheus has generated an extensive number of exporters that can be used to collect telemetry from multiple sources in a uniform time-series format. Prometheus exporters are http-based services that expose time series as text documents that can be periodically scraped by Prometheus. If not available yet, a dedicated Prometheus exporter can be developed to collect metrics from the RAN DB.
- Thanos (https://thanos.io/): Used as the long-term storage for the Prometheus telemetry. Thanos is designed to aggregate and consolidate short-lived data stored in Prometheus cluster-stores, as well as to expose a long-term Prometheus-compatible interface to the data. Internally, Thanos can split time-series queries covering long-term time ranges into subqueries that are served both from the Thanos long-term repository, and from the distributed short-term data repositories from multiple Prometheus instances. Therefore, Thanos produces aggregated results including long-lived and short-lived data from a single interface. Prometheus instances registered in a Thanos service include a Thanos-sidecar responsible to push long-lived data as well as to serve queries issued from the Thanos.

- MinIO (https://min.io/): Acts as a S3-compatible data storage used by Thanos to store data blocks.
- Grafana (https://grafana.com/grafana/): To analyze and visualize application, service, and infrastructure telemetry (KPIs) in real time.

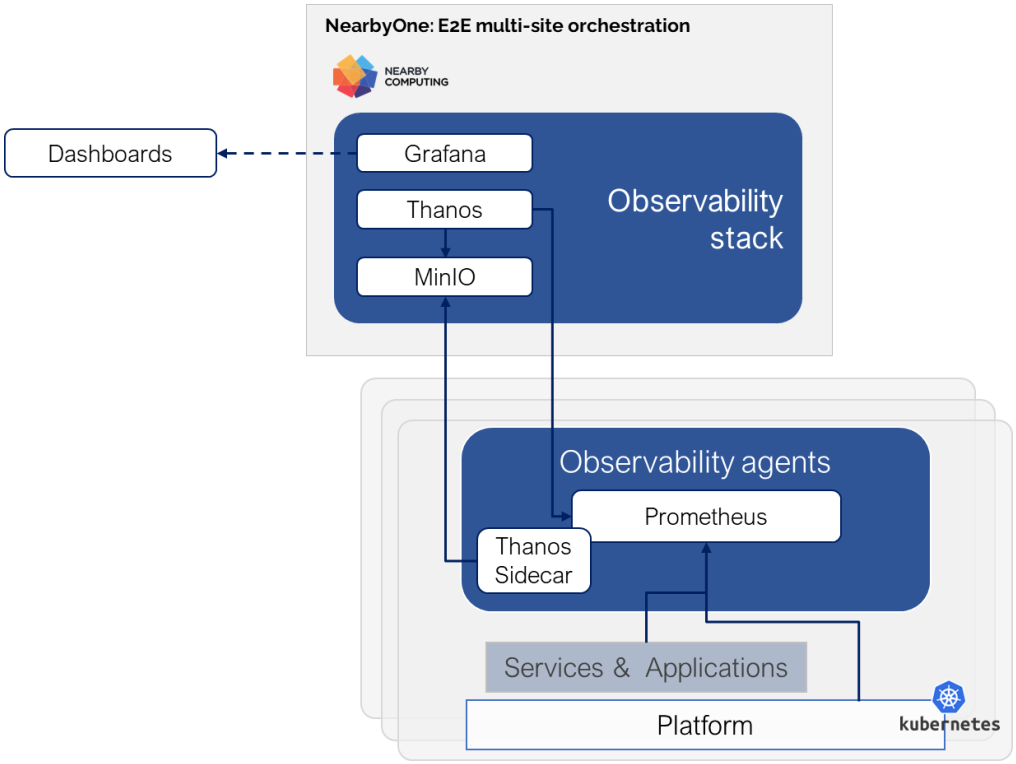

Figure 23 NearbyOne telemetry stack

7 Data Analytics Layer Architecture

In a 5G architecture, the Data Analytics layer plays a pivotal role in extracting insights, processing the amounts of data generated by the network and the vehicle and utilizing analytics to enhance network performance and user experience. Based on the knowledge extracted by the condition monitoring service and with the help of appropriate visualization tools and platforms, instructive and actionable insights can be derived in real-time towards an enhanced end-to-end performance, e.g., flag whether a fault has happened and diagnose its type in event-detection operations.

The responsibilities that are included in the scope of the SUCCESS-6G-DEVISE project are limited to the following:

Data Processing and Analysis: The Data Analytics layer handles the massive volume of data generated by the IoT sensors of the car, base stations, and network functions. It processes this data in real-time or near-real-time to derive meaningful insights.

Predictive Maintenance and Fault Detection: Using machine learning algorithms and predictive analytics, this layer identifies potential sensor issues or equipment failures before they occur. It enables proactive maintenance, reducing downtime and enhancing network reliability [1].

Security and Anomaly Detection: Monitoring network traffic for unusual patterns or security threats, the Data Analytics layer assists in detecting and mitigating cyber threats. It identifies anomalies in data transmission or user behavior that could indicate security breaches, allowing for timely responses.

The following sections will provide an insight into the different functionalities provided by the layer and required to the proper function of the same.

7.1 Data extraction and observation

As related in the previous section, the Data Analytics layer is responsible for:

1. Exploratory data analysis
2. Corresponding visualizations

Data consists of gathered measurements, model predictions and systems measurements, i.e. systems telemetry, device/edge device power consumption estimates, CPU/RAM/GPU usage, etc. The analytics layer will be implemented as a part of NearbyOne Observability stack described in section 6.2, whose main components are:

- Custom Grafana dashboards, for online measurements and predictions visualization.
- Custom Grafana reports, which are automatically generated reports for performance summary and data analysis.
- Prometheus, which will be in charge of the export of system metrics.
- Kepler (Kubernetes-based Efficient Power Level Exporter) is a Prometheus exporter. It uses eBPF to probe CPU performance counters and Linux kernel tracepoints.
- Thanos, which will be used to store measurements and predictions.

7.2 MLOps Architecture

As each user story shares ML Operation flow, it is described in Figure 24 and can be considered common to the user stories regarding the obtention of the used ML model. Figure 24 represents the different levels of maturity of the MLOps flow, that will be achieved throughout the project.

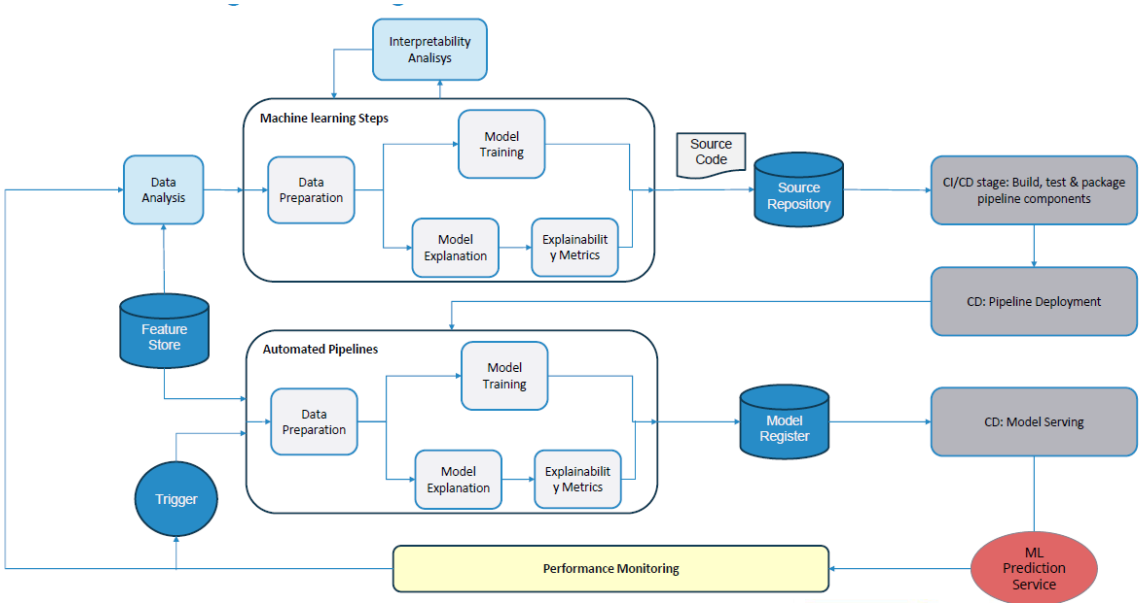

Figure 24 Suggested MLOps flow for the vehicular condition monitoring and fault provisioning

The preparation of the ML models starts with data gathering. Being it for the first user story, where we will try to infer missing or wrong information, or for the second story, on which we will try to detect and mitigate security issues.

-**Machine Learning Steps**: This stage represents the basic level, where all ML processes, from data collection and preparation to model training and validation, are manually executed. Consequently, it is termed as 'no MLOps.' We will use rapid application development tools like Jupiter Notebooks to construct learning models. The transitions between stages are manual and driven by source code iteratively developed until an executable model meeting system requirement is achieved.

-**Training Pipeline Automation**: This level introduces continuous model training, automating the training steps. When new data profiles emerge, it triggers the model retraining process. This cycle encompasses data and model validation phases to ensure a consistent delivery of learning models. Here, two new components are introduced: a feature store acting as a centralized feature repository enabling access to new features, and machine learning metadata storing ML pipeline execution information.

-**Model Training, Evaluation, and Saving**: This module segregates processes for continuous automatic model training using processed data.

The ML model training involves local optimization, while model explanation includes an explainer (either attribution-based or perturbation-based).

Evaluation of RL reward and interpretability metrics triggers model retraining upon performance decline with new training data arrivals.

A feedback loop between explainer and model feeds metrics to the model optimizer, enabling explainability-aware learning.

Successful models meeting performance targets are registered in the model registry for future production deployment.

-**Continuous Integration and Delivery Pipeline Automation**: This level introduces a comprehensive CI/CD system to enable reliable and swift learning model deployment.

As the calculations required to calculate the different models to apply to the three user stories might be computationally expensive it will be decided, during the WPs 4-5, if this MLOps services will be

allocated in the private cloud present in the circuit or in a public cloud instance with the necessary hardware behind.

7.3 ML design steps

Deepening in the ML model design steps, we can include the following actions in each of the blocks, as shown in Figure 25:

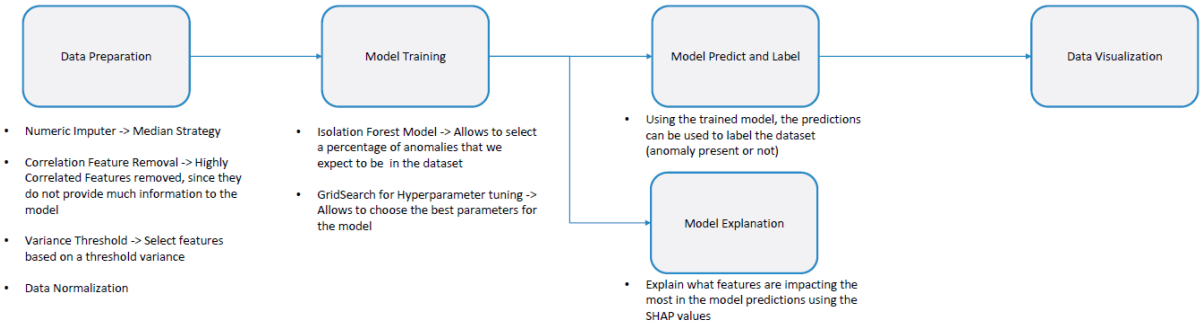

Figure 25 ML design steps

8 Local Cloud Architecture Description

SUCCESS-6G local cloud provides a system of virtual machines that support different services of the 5G network, besides MEC applications for the end2end. The local cloud is supported by the ICT infrastructure located in the Control Room of Castelloli Mobility Lab. It is composed basically of virtualized machines running on a server. The server is a Lenovo ThinkSystem SR650 (shown in Figure 26), that provides the high-performance computing needed to support the local cloud and data workload management. It is a 2RU units HW server, based on 2nd Gen of Intel Xeon processors.

Figure 26 Server Lenovo SR650

The VM Sphere solution facilitates the efficient deployment and management of the virtual machines that will be hosted by the server. The 5G-Core of the private network is a service-based solution that runs in the Local Cloud. Besides, local cloud also provides the infrastructure for the virtualized MEC services. It contains the distributed UPF instances and can support additional virtualized MEC applications if needed. Services will be deployed in two separated Lenovo SR-650 servers (virtualized) placed in the same rack as depicted in Figure 13 and Figure 17.

To deploy the different services in the virtualized SR650, both servers have installed vSphere (vmware), a software that allows to create different virtual machines and allows to virtualize the resources. The characteristics of the virtual machine for the different use cases can be configured for each VM. To get access to this VM and the other equipment, a VPN (Virtual Private Network) will be granted allowing access to the different VLANs.

9 Workflows

For each use case and user story, as defined in Deliverable E5 [3], the entire process is meticulously defined using a set of sequence diagrams, which provide a clear and detailed visual representation of the interactions between the various components defined in the SUCCESS-6G-DEVISE architecture. These diagrams serve as a valuable tool for understanding the workflow and ensuring that all components interact seamlessly to provide a robust and efficient solution.

9.1 Use case 1 Workflow: Vehicular condition monitoring and fault provisioning

9.1.1 Workflow: Vehicular condition monitoring with security guarantees

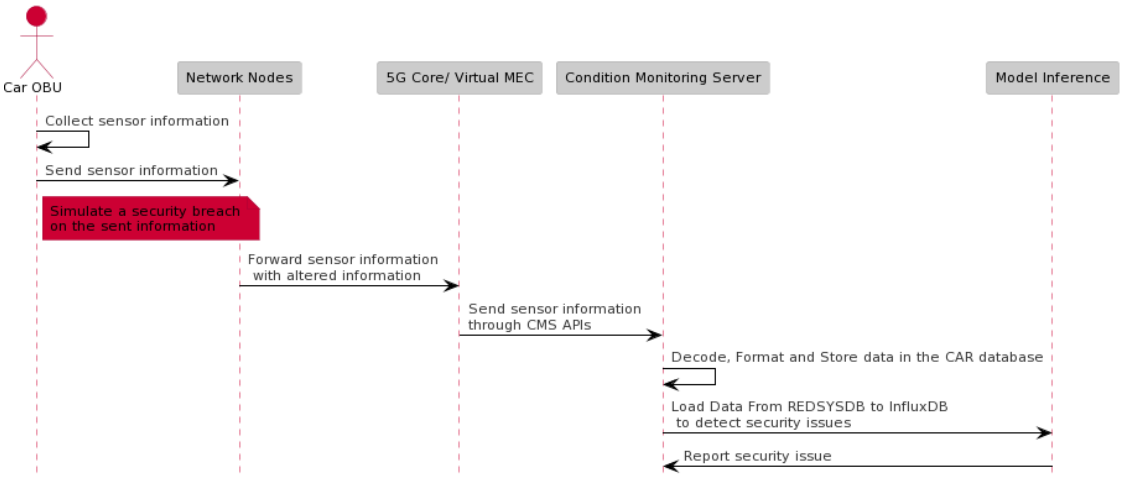

Figure 27 Sequence diagram for vehicular condition monitoring with security guarantees

As shown in Figure 27:

1. Data is captured from the OBD port where ECU information from sensors installed in the vehicle is available.

2. Transmission to the edge monitoring infrastructure takes place with the aid of a vehicular OBU which fuses aggregated information from various sensors and a C-V2X roadside infrastructure comprising mobile radio stations and/or road-side units (RSU).

3. Measurement streams are aggregated and processed at the edge monitoring units. Aggregated information can be directly or indirectly used for downstream tasks: i) quantification of the impact of threat vectors injecting falsified information in supervisory data; ii) detection of injected false data with the aid of AI/ML techniques to ensure trustworthy connectivity; iii) enforcement of security policies and development of defensive measures towards secure condition monitoring by empowering a fully new breed of autonomic cyber-capabilities, e.g., self-protection, self-healing, which increase resiliency to attacks.

4. Based on the knowledge extracted and with the help of appropriate visualization tools and platforms, instructive and actionable insights are derived by the maintenance team towards an enhanced end-to-end performance, e.g., flag whether a fault has occurred and diagnose its type in event-detection operations. The selection of appropriate security countermeasures for trustworthy decision-making is also performed.

5. Appropriate actions, e.g., alerts, modification of sensor reporting frequency, are

communicated back to the vehicle as a response to the automated supervision of measurement flows. Attack/threat mitigation strategies should have minimum impact on the underlying communication.

9.2 Use case 2 Workflow: Over-the-air vehicular software updates

In this section, we present the sequence diagram for the proposed architecture in Figure 3 which details over-the-air vehicular software updates with security guarantees.

9.2.1 Workflow: Over-the-air vehicular software updates with security guarantees

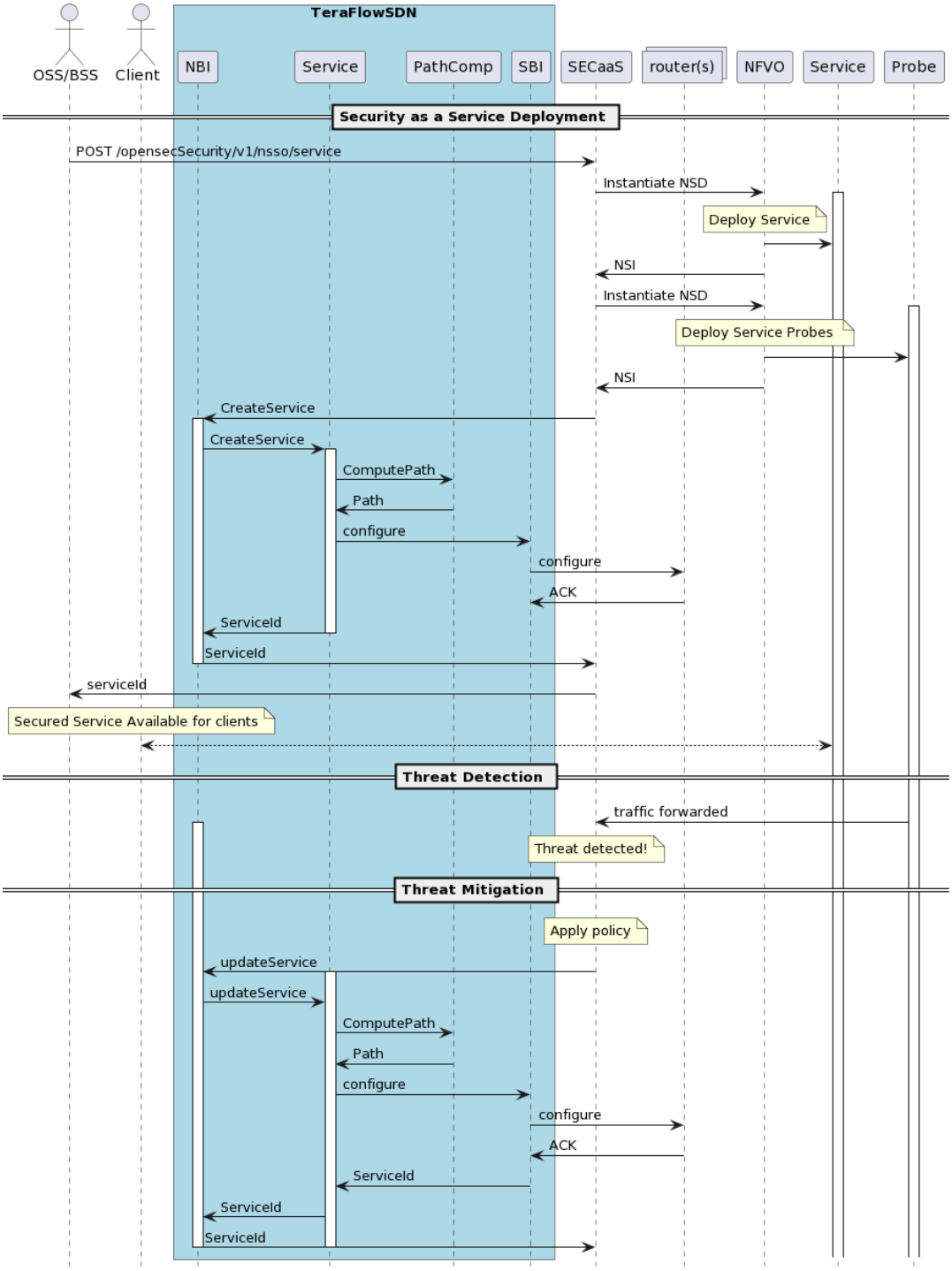

Figure 28 Sequence diagram for Over-the-air vehicular software updates with security guarantees

The provided Figure 28 delineates the orchestrated deployment of Security as a Service within the

TeraFlowSDN framework, elucidating the interaction dynamics among key actors and components. The diagram captures two key phases: "Security as a Service Deployment" and subsequent processes related to "Threat Detection" and "Threat Mitigation."

Security as a Service Deployment

The sequence initiates with the Operations Support Systems/Business Support Systems (OSS/BSS) actor initiating a security service deployment by making a POST request to the Security as a Service (SECaaS) module. This triggers the instantiation of a Network Service Descriptor (NSD) through the Network Function Virtualization Orchestrator (NFVO), leading to the deployment of the security service and associated probes. The NFVO interacts with both the SECaaS module and the deployed Service, ensuring the establishment of Network Service Instances (NSIs).

Subsequently, the OSS/BSS actor communicates the instantiation of the security service to the client through the transmission of the serviceId. This information exchange is visually represented by bidirectional arrows between the client and the Service component. The client is thereby notified of the availability of the secured service.

Threat Detection

The diagram transitions into the phase of threat detection, where the Probe component forwards traffic information to the SECaaS module. The notation underscores the detection of a threat, and this pivotal information is communicated between the Probe and SECaaS components.

Threat Mitigation

Upon threat detection, the SECaaS module undertakes threat mitigation measures. This involves the application of security policies, as indicated by the "Apply policy" notation. The orchestration of these mitigation actions is captured in the subsequent sequence. The NFVO is activated, and the SECaaS module communicates with the Northbound Interface (NBI) to update the existing service. This update triggers a series of actions, including the computation of a new service path by the PathComp component, device configuration via the Southbound Interface (SBI), and acknowledgment exchanges between routers and the SBI. The Service component, in turn, communicates the updated ServiceId back to the NBI and subsequently to the SECaaS module.

In a concise summary, this diagram meticulously documents the sequential deployment of Security as a Service, threat detection, and subsequent mitigation measures within the TeraFlowSDN architecture, providing a clear visual representation of the intricate interactions and processes involved in ensuring the security and resilience of network services.

10 System Requirements

This section describes the requirements that drove the overall SUCCESS-6G-DEVISE architecture design both from a functional and non-functional point of view. Functional requirements define a specification of a behavior between inputs and outputs (function) of a system or its components. On the other hand, non-functional requirements define specification criteria that can be used to evaluate the operation of a system. Non-functional requirements impose constraints on the design or implementation of system such as performance, security, or usability requirements.

System requirements are expected to provide guidance to the development of WP3-5 technical solutions. We provide the details in the following.

10.1 Functional Requirements

Functional requirements refer to a set of specifications that outline the tasks that a system must be capable of performing, as well as the way in which those tasks should be carried out. These specifications define the functions, features, and abilities that are essential for a system or software application to satisfy the requirements of its users.

The functional requirements are listed as follows:

Functional Requirements	Description (Optional)
The OBU must be capable to operate in Castellolí test bed	5G SA and N77 band
The OBU must be capable to read sensor data from internal vehicle CAN bus	OBD2 connector, with CAN baudrate 500kbps and no FD
The OBU must be capable a high accuracy position in the test bed	<1m error
The OBU must be capable to receive over-the-air system updates for itself or other OBUs of the vehicle	
The OBU must be capable to secure the communication channel between the OBU and the infrastructure	TLS 1.3 encryption
The condition monitoring service needs to interact with the data analytics layer	Measurements and predictions will be exchanged for robust model building

	(tracking of the ML model experiments) and outlier detection
The condition monitoring service needs to be integrated with the NBC orchestrator (containerized) and run statically/dynamically at multiple edges.	Service to be included in the marketplace for rapid service deployment through the NearbyOne dashboard
UPF deployment should be performed via the NearbyOne orchestrator	For network reconfiguration purposes

Table 5 Functional requirements

10.2 Non-Functional Requirements

Non-functional requirements are a set of specifications that detail the characteristics, attributes, and qualities of a system or software application, as opposed to its specific functional capabilities. These requirements pertain to the general performance, quality, and usability of the system or software application, rather than its functionalities and features.

The non-functional requirements are listed as follows:

Non-Functional Requirements	Description (Optional)
Orchestrated use-case applications	NearbyOne orchestrated services or applications are cloud-native applications packaged with Helm[2].
Registry for orchestrated use-case applications	Orchestrated application artifacts and manifests (e.g., Helm Charts, Docker containers) are available in a registry accessible by NearbyOne.
Visualization capabilities of the data analytics layer	Support tailored queries, provide charting capabilities and alerts for measurement live charts and notifications

[2] https://helm.sh/docs/helm/helm/

The network infrastructure shall warrant measurable level of availability of its services to the relevant stakeholders.	
The network infrastructure shall ensure the necessary network capacity and network resources for the critical operations of the vehicular services.	

Table 6 Non-functional requirements

11 Capabilities and interface definition

11.1 Vehicle

Service	Description	Visibility	Potential Consumer
Vehicle sensor data	Vehicle sensor data is available in the internal CAN bus	Can be accessed reading physically the CAN bus with a CAN transceiver	OBU
OBU	Hardware device installed in the vehicle capable to read sensor data from the internal CAN bus a send this information to the MEC	The OBU will provide the data via TCP/IP in the local net provided by Cellnex	MEC

Table 7 Capabilities and interface definition for vehicle

11.2 MEC

Service	Description	Visibility	Potential Consumer
Vehicle Data Monitoring	The vehicle monitoring service is responsible for decrypting, decode, time-tag and process messages received from the vehicle	Readable time tagged data in events format	Transactional database
Transactional database	This database store events while maintaining data integrity in multi-access environments	REDIS database	Generic Database

Table 8 Capabilities and interface definition for MEC

11.3 Data Analytics

Service	Description	Visibility	Potential Consumer
Generic database	This database stores the post-processed data of the sensors	External	MEC

Table 9 Capabilities and interface definition for data analytics layer

12 Conclusions

This deliverable focuses on the architecture required for the development of the different capabilities needed to fulfil the vehicular use cases that will be targeted by SUCCESS-6G-DEVISE project. It presents the initial set of capabilities that will be covered by each module of the project, which will be developed by project partners to address the technical challenges associated with them. Also, each SUCCESS-6G-DEVISE user story has been described via workflows indicating a sequence of steps that will be held in the future WPs.

References

[1] P. Mulinka et al., "Information Processing and Data Visualization in Networked Industrial Systems," 2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Helsinki, Finland, 2021, pp. 1-6, doi: 10.1109/PIMRC50174.2021.9569603.

[2] Helm - The package manager for Kubernetes, https://helm.sh/docs/topics/charts

[3] SUCCESS-6G: DEVISE https://success-6g-project.cttc.es/images/deliverables/E5-DEVISE-2.pdf