PROJECT "LOCUS": LOCalization and analytics on-demand embedded in the 5G ecosystem, for Ubiquitous vertical applicationS

Grant Agreement Number: 871249
(https://www.locus-project.eu/)

DELIVERABLE D6.2

"Network management, network-assisted self-driving vehicles, people mobility and flow monitoring applications, integrated with geolocation mechanisms"

Deliverable Type:	R/DEM
Dissemination Level:	Public
Contractual Date of Delivery to the EU:	28/02/2022
Actual Date of Delivery to the EU:	10/03/2022
WP contributing to the Deliverable:	WP6 – Proof of Concepts
Editor(s):	INCE, Athina Ropodi
Author(s):	VIAVI, Takai Eddine Kennouche, Karen Boulos UMA, Emil J. Khatib TEI, Marzio Puleri, Stefano Stracca INCE, Aristotelis Margaris, Yannis Filippas, Kostas Tsagkaris

	NEC, Gürkan Solmaz NXW, Elian Kraja, Nicola Venturi, Giacomo Bernini, Michael De Angelis CNIT, Gianluca Torsoli, Flavio Morselli, Andrea Conti, Domenico Garlisi OTE, Maria Belesioti ORA, Sana Ben Jemaa
Internal Reviewer(s):	IBM, Faisal Ghaffar
Short Abstract:	The goal of this deliverable is to provide details on the Proof-of-Concept applications that will be powered by the LOCUS geolocation data and analytics, describe the Proof-of-Concept environment and relevant testbeds, explain the use cases' execution framework, and sketch the implementation of the applications and their main components.
Keyword List:	PoC, Use Case, application, geolocation, analytics, Testbed

Executive Summary

This deliverable documents the work being done in the context of Work Package 6 for the realization of the Proof of Concepts (PoCs) of the LOCUS project. More specifically, it details the PoC environment, i.e. System Architecture aspects and the relevant Testbed capabilities - including technologies and resources. It additionally provides specification of applications that will be powered by the LOCUS geolocation data and analytics of the LOCUS platform and sketches the implementation of PoC applications and their main components. While the PoCs are categorized under three main categories, namely "Network Management based on Location Information", "Network-assisted Self-driving Objects" and "People Mobility & Flow Monitoring", it should be noted that –as described in this document- the LOCUS Consortium has further extended the initial number of PoC use cases. More specifically, it has been decided to include additional LOCUS functions for the PoCs and provide added and enhanced scenarios, in line with the work taking place in relevant work packages- so as to offer demonstrations that would best represent the range of the LOCUS system capabilities.

VERSION CONTROL TABLE			
VERSION N.	**PURPOSE/CHANGES**	**AUTHOR(S)**	**DATE**
0.0	ToC	See list above	12/01/2022
0.1	Finalized ToC	INCE, NXW, NEC	24/01/2022
0.11	Minor updates in ToC	UMA	27/01/2022
0.2	First round of contributions	OTE, INCE	14/02/2022
0.3	Second round of contributions	NEC, CNIT, VIA, NXW, TEI, UMA, INCE	21/02/2022
0.4	Contributions after internal review, added text, references	UMA, ORA, VIA, NXW, INCE	25/02/2022
0.5	Final updates in text, list of abbreviations & references. Version ready for internal review	NEC, CNIT, TEI, INCE	28/02/2022
0.6	Reviewed document	IBM	04/03/2022
1.0	Correction and updates based on review – Ready for submission	INCE	09/03/2022
1.1	Final revision by Coordinator	Nicola Blefari Melazzi	10/03/2022

INDEX

1 Introduction

1.1 Introduction and Audience

In the context of the LOCUS project, Work Package (WP) 6 “Proof of Concepts” refers to the definition and requirement specification of all use case (UC) scenarios leading to the integration of the various LOCUS components and ultimately the overall system integration, based on the finalized LOCUS Platform architecture described in Deliverable 2.5 [1]. The Proof of Concepts (PoCs), described also previously in Deliverable 6.1 [2], are divided in three main categories corresponding to the relevant WP Tasks 6.1-6.3. In detail, the PoCs based on the project’s document of work (DoW) are the following:

1) Network Management based on Location Information (PoC 1)
2) Network-assisted Self-driving Objects (PoC 2)
3) People Mobility & Flow Monitoring (PoC 3), subdivided in two sub cases:

   3a) Flow tracking in densely populated indoor environments for smart retail and venue management

   3b) Crowd mobility analytics using wireless and auxiliary sensors

However, it should be noted that LOCUS partners have further extended the PoCs’ definitions, describing a number of additional PoC Us in this deliverable, i.e. Transportation, Collision Detection for Self-Driving Cars, Network Planning and Redesign Using Hybrid Spatial Network Clustering.

In addition to the above, according to the roadmap of WP6 activities, the involved demonstrators of the various use cases have already begun development of their respective components in an internal development instance. In this document, the progress for the migration from local instances to the final testbeds for the demonstrations is also reported.

This document provides a description of the three PoC groups, including details of their implementation, and the realization of relevant use cases. As this is a public deliverable, it is intended for the LOCUS consortium members, the EU commission, as well as the general public.

1.2 Document Structure

Deliverable 6.2 “Network management, network-assisted self-driving, people mobility and flow monitoring applications, integrated with geolocation mechanisms” is structured as follows:

- Section 1 includes a high-level introduction to the PoCs and the deliverable objectives.
- Section 2 describes the PoCs execution framework, i.e. System Architecture aspects and related infrastructure in the two testbed sites, namely OTE and UMA testbeds. It

further expands on the efforts and progress of migrating from local instances/ smaller scale deployments in various partners' premises and emulation-based analysis to the UMA and OTE testbeds where the LOCUS demonstration platform will be deployed.

- Section 3 provides the status and any progress/ updates with respect to the PoC use cases already described in D6.1 [2], including: (i) service design, i.e. the design of the services that are described in the various use cases and are to be deployed for the final demonstrations; (ii) views' analysis, presenting the views/ user interface per use case and (iii) scenarios, i.e. storylines that are going to be displayed.
- Lastly, Section 4 describes the main components and implementation aspects of additional PoC UCs, in terms of UC services, user interfaces and scenarios to be demonstrated.

1.3 List of Abbreviations

Table 1.1 Abbreviation List

ABBREVIATION	FULL NAME
3GPP	3rd Generation Partnership Project
AI	Artificial Intelligence
AMF	Access and Mobility Management Function
AMQP	Advanced Message Queuing Protocol
API	Application Program Interface
AUSF	Authentication Server Function
CNF	Containerized Network Function
CMU	Compact Mobility Unit
DoW	Document of Work
eCNS	evolved Core Network Solution
FQDN	Fully Qualified Domain Name
FTM	Fine Time Measurement
(G)UI	Graphical User Interface
HSS	Home Subscriber Server
k8s	Kubernetes
KNF	Kubernetes Network Function
KPI	Key Performance Indicator
LBS	Localization Base Service

LMT	Local Management Terminal
LTE	Long-Term Evolution
MANO	Management and Orchestration
ML	Machine Learning
MME	Mobility Management Entity
NFVI	Network Functions Virtualization Infrastructure
NSD	Network Service Descriptor
OSM	Open-Source MANO
PCRF	Policy and Charging Rules Function
P-GW	Packet Gateway
PID	Proportional Integral Derivative
RAN	Radio Access Network
RAT	Radio Access Technology
RSRP	Reference Signal Received Power
RSSI	Received Signal Strength Indicator
S-GW	Serving Gateway
SI	Soft Information
SMF	Session Management Function
SON	Self-Organizing Networks
PoC	Proof of Concept
UC	Use Case
UDM	Unified Data Management
UE	User Equipment
UMA	University of Malaga
UPF	User Plane Function
UWB	Ultra-Wideband
VM	Virtual Machine
VNF	Virtual Network Function
VNFD	Virtualized Network Function Descriptor
WP	Work Package

1.4 List of Terms

In this section, the basic terminology used throughout the text is clarified briefly below:

- **Proof of Concept (PoC)**: The term refers to the demonstrations that will take place in the context of WP6 and their purpose is to display the most representative aspects of the project's work and specifically the LOCUS Platform and its related capabilities.
- **Use case (UC)**: A Use Case refers to one or more application scenarios, describing representative uses of the LOCUS Platform related to one or more of the following: (a) Localization techniques; (b) Smart Network Management, related to telecom operators; and (c) New Services, related to 3rd party/ vertical uses. UCs were first detailed in WP2 and their relevant LOCUS localization and analytics functions were developed in the context of WP3-WP5. A PoC may demonstrate one or multiple UCs.
- **Scenario**: The outline of a specific application of the LOCUS Platform in the context of the PoCs and with respect to the existing UCs. One UC may have one or more scenarios, also mentioned as storylines. Also, a scenario may be a simplified version or a partial aspect of a UC.
- **Architecture**: All the aspects/ specifications related to the LOCUS Platform and its ecosystem, including main components blocks, interfaces, and specific technologies, implementation aspects and execution framework.
- **Infrastructure**: Related to the resources (hardware and software) and relevant technologies concerning the PoC environment, the testbeds/ simulators used for the realization of the PoCs and their capabilities.

1.5 Table Index

1.6 Figure Index

2 Proof of Concept Testbed Infrastructure

2.1 System Architecture Overview

The LOCUS Platform exposes functionalities, services and interfaces towards Smart Network Management (i.e. catering to the telecom operators' and network owners' need for intelligent management of their network) and New Services/3rd party applications, meaning novel and improved applications that may be offered by verticals. The goal is to enable seamless integration of location-awareness in the 5G ecosystem. For this, the system architecture, as described in D2.5 [1], empowers functionalities and implements design choices that allow for: (a) the flexible deployment and training of Artificial Intelligence (AI)/ Machine Learning (ML) powered services and functions across the network; and (b) the extraction and exploitation of geolocation information to fully benefit across network elements from modern AI/ML approaches.

For this reason, the architecture accounts for:

- scalable and efficient data persistence and data movement;
- lifecycle management of ML pipelines;
- decoupling of architecture design and implementation to allow developers to implement solutions using the best tools for the task;
- containerization and microservice-based architecture, and the possibility to build services and pipelines;
- support of privacy and security functions.

The PoCs described in this document aim at implementing a set of functionalities that demonstrate the inter-working of the developed LOCUS functions and Application Programming Interfaces (APIs) based on the LOCUS architecture design described in Deliverable 2.5 [1] for the empowerment of various location-aware scenarios.

In the following sections the PoC testbeds at OTE and UMA premises, as well as the relevant platform architecture are described.

2.2 OTE Testbed Infrastructure

2.2.1 *Technologies*

The OTE testbed setup is based on Ubuntu Server 18.04 vanilla OS [3], with Queens OpenStack version [4]. The testbed to be employed for the LOCUS project PoC purposes currently consists of one Controller node and three Compute nodes (see also section 2.2.2, Table 2.1). The currently deployed setup is presented in Figure 2.1.

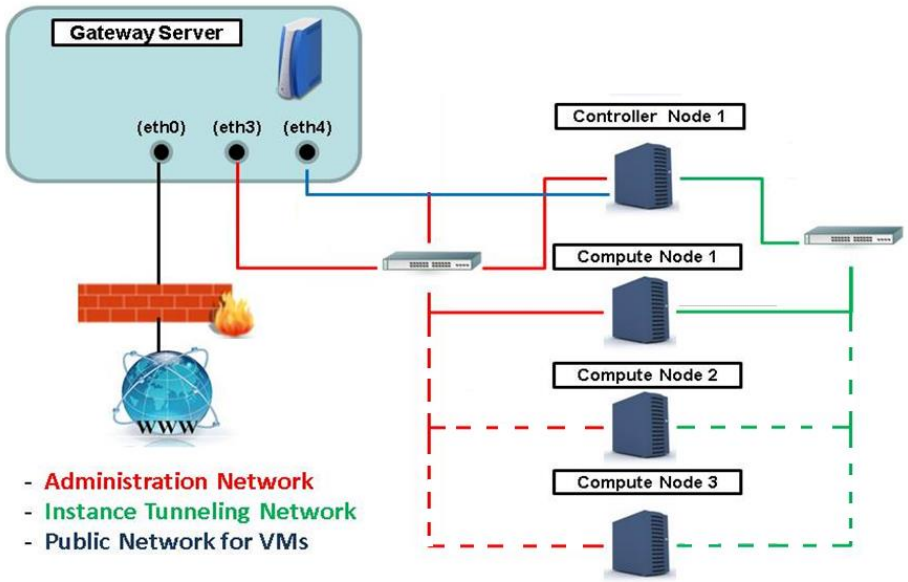

Figure 2.1 Testbed deployment at OTE premises

In more detail, the following components are installed in Openstack [5]:

- Keystone, for API client authentication, service discovery, and distributed multi-tenant authorization;
- Glance, providing the compute image repository. All compute instances launch from glance images;
- Nova, a cloud computing instance controller;
- Neutron, responsible for provisioning the virtual or physical networks that compute instances connect to on boot;
- Cinder, which manages volumes, and interacts with OpenStack Compute to provide volumes for instances;
- Horizon, the official web user interface (UI) for the OpenStack Project

2.2.2 *Resources*

Based on the LOCUS testbed estimated dimensioning to support the PoCs, the following hardware resources have been assigned. This estimation has been performed based on the amount of resources required to run each PoC, in terms of UC functionalities to be dynamically deployed as virtualized functions, orchestrated by the LOCUS Management and Orchestration Platform. In addition, the dimensioning considered the resources required to run the LOCUS Platform services as well. The calculations take into consideration the possibility to have the execution of at least two simultaneous PoCs. More details on the purpose of the resources assigned and described here can be found in the section describing the Virtualization Platform architecture (section 2.2.4).

In terms of virtualized resources, the testbed is deployed in three compute nodes managed by *locuscontroller*. The compute nodes are: (a) "*EGDE Cloud",* dedicated to the edge domain; (b) *"LOCUS Platform"*, hosting all the LOCUS Platform components; and (c) -referring to the

Network Functions Virtualization Infrastructure- "*NFVI*" (i.e. to be used for the instantiation of the LOCUS functions). The details about the virtual resources each of them uses, as well as the resources used up to now are provided in Figure 2.2. As shown, not all resources are utilized at this moment, as (a) not all components are currently deployed and (b) resources are assigned under the assumption that more than one scenario will be up and running at different times.

Table 2.1 OTE testbed's hardware resources assigned to LOCUS PoCs

Hostname	Model	#Cores	RAM (GB)	HD	NICs
locuscontroller	Proliant ML110 Gen10	16	48	1TB SSD + 2X2 TB SATA	ens1f0, ens4f0
locuscompute1	Proliant ML350 Gen9	88	256	2x500GB NVME + 8X2 TB SATA	ens1f0, ens3f0
locuscompute2	Proliant DL380 Gen10	88	256	2x500GB NVME + 3x2TB SATA	ens1f0, ens3f0
locuscompute3	Proliant DL380 Gen10	88	256	2x500GB NVME + 3x2TB SATA	ens1f0, ens3f0
Switch	HPE Flexfabric 5700	-	-	-	13-16, 18,20,30,31

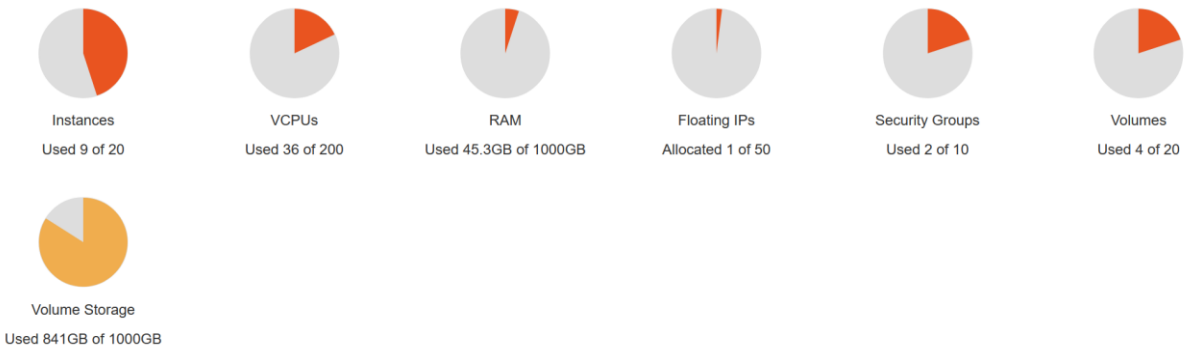

Figure 2.2 Overview of the virtualized resources used for the purposes of the LOCUS Testbed

Edge Cloud compute node

Edge Cloud contains three Virtual Machines (VMs) as presented in Figure 2.3. It is a cluster consisting of three nodes, a master node and two worker nodes, catering to the edge infrastructure, while an orchestrator is necessary to manage the compute, storage, and networking resources (see Figure 2.9). The related virtual resources are also shown in Figure 2.4.

☐	Instance Name	Image Name	IP Address	Flavor
☐	kube-worker-2	Ubuntu-18.04	172.16.12.32	k8s_worker
☐	kube-worker-1	Ubuntu-18.04	172.16.12.51	k8s_worker
☐	kube-master	Ubuntu-18.04	172.16.12.65	k8s_master

Figure 2.3 Edge Cloud compute node's virtual machines

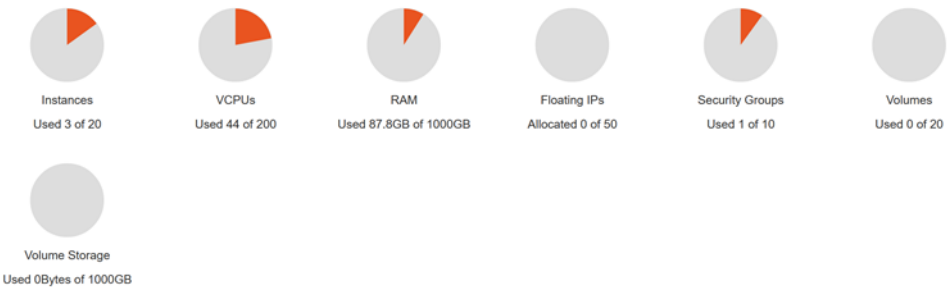

Figure 2.4 Edge Cloud compute node's virtual resources

LOCUS Platform compute node

LOCUS Platform contains the following VMs (Figure 2.5), while the virtual resources consumed for LOCUS purposes are presented in Figure 2.6. The main components found here are the LOCUS management and orchestration module (*"OSM-R10"* in Figure 2.5), a container registry component ("*Harbor-Registry*") that stores and allows distribution of LOCUS functions, a

message broker *("DATA-MOVEMENT")* for publishing and consuming data, a VM with installations for the LOCUS Persistence Module component *("PLACEMENT")*, and a DNS service (*"DNS"*) to easily identify the different LOCUS platform services and functions.

☐	PLACEMENT	Ubuntu-20.04	172.16.12.33	PLACEMENT
☐	DATAMOVEMENT	Ubuntu-18.04	172.16.12.27	DATAMOVEMENT
☐	DNS	Ubuntu-18.04	172.16.12.25	m1.small
☐	Harbor-Registry	Ubuntu-18.04	172.16.12.26	REGISTRY
☐	OSM-R10	Ubuntu-18.04	172.16.12.43	OSM-FLAVOR

Figure 2.5 LOCUS Platform compute node's virtual machines

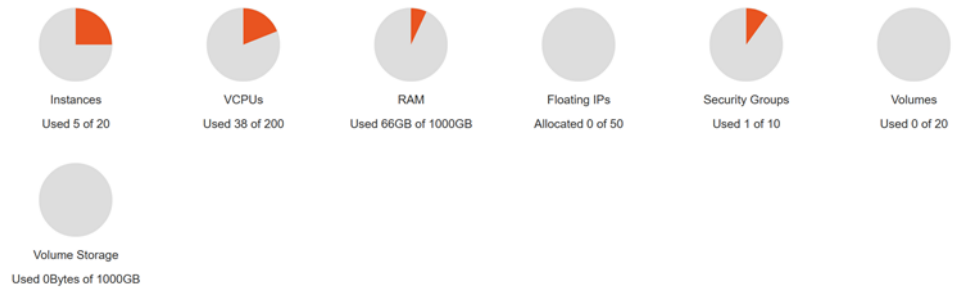

Figure 2.6 LOCUS Platform compute node's virtual resources

NFVI compute node

NFVI is the last compute node of the OTE/LOCUS testbed. Currently it does not host any VMs and does not consume any resources as it is depicted below (Figure 2.7), since not all components are currently deployed.

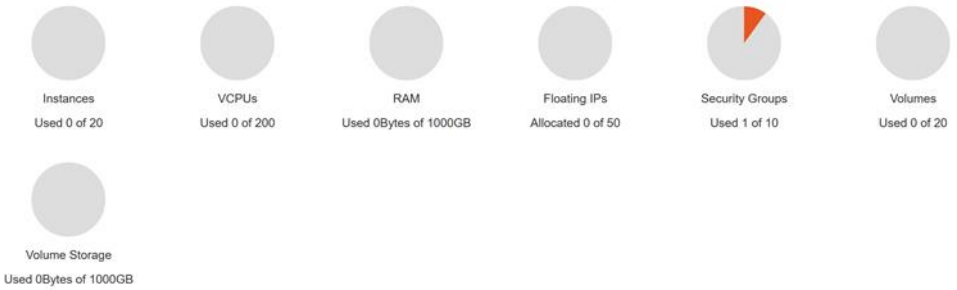

Figure 2.7 NFVI compute node's virtual resources

2.2.3 *Networking*

The LOCUS Provider Network is 172.16.12.0/24 and the Reserved Management IPs are 172.16.0.41-49. More specifically:

- Management IP (locuscontroller): 172.16.0.44

- Management IP (locuscompute 1): 172.16.0.43
- Management IP (locuscompute 2): 172.16.0.45
- Management IP (locuscompute 3): 172.16.0.46

A high-level network topology is depicted below (Figure 2.8):

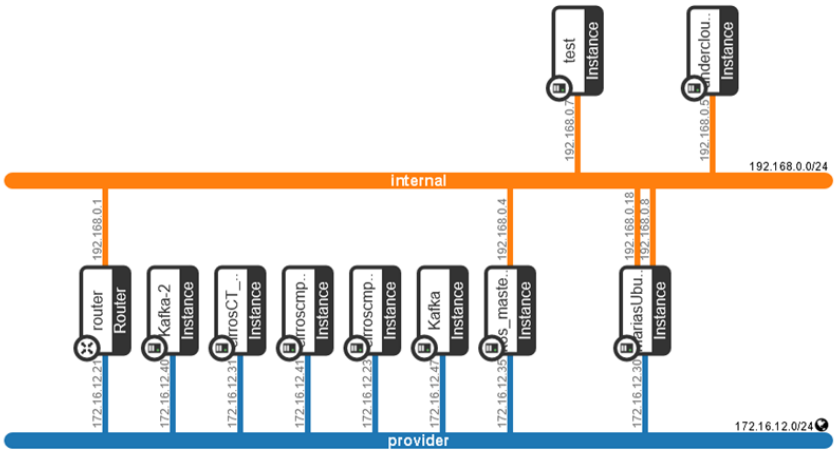

Figure 2.8 OTE testbed network topology

2.2.4 *LOCUS Virtualization Platform*

The LOCUS Virtualization Platform is hosted in the OTE infrastructure, described in Section 2.2. The Virtualization Platform is implemented leveraging different virtualization technologies, allowing the execution of different PoC scenarios. The initial version of the LOCUS Virtualization Platform, designed and preliminary deployed in an early release in the lab of the Nextworks partner, is described in deliverable 4.3 [6]. The final version of the virtualization platform will be described in detail in deliverable 4.4 [7].

Given the diversity of the LOCUS functionalities to be hosted on the LOCUS Platform, the virtualization platform allows the execution of cloud-native applications both across edge and core locations, depending on the needs.

Figure 2.9 depicts the architecture of the LOCUS Virtualization Platform.

As previously described in deliverable 4.3 [6], both edge and core virtualized infrastructures use different technologies. The core infrastructure is based on OpenStack [8], a cloud operating system able to control large pools of compute, storage, and networking resources throughout a datacentre. On the other hand, the edge infrastructure is based on Kubernetes (k8s) [9], an open-source platform for managing containerized workloads and services.

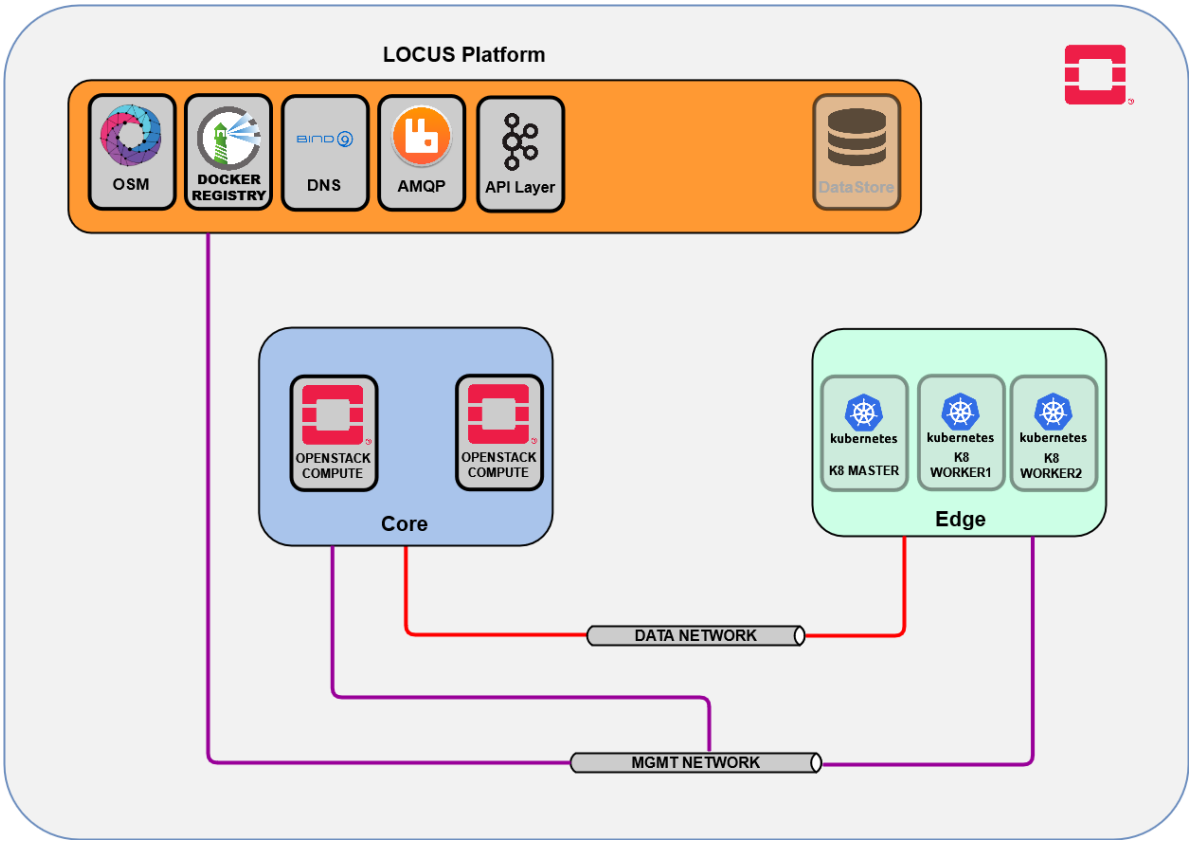

Figure 2.9 LOCUS Virtualized Platform Architecture

Along with the virtualized infrastructure, the LOCUS Platform also provides a set of platform components that enable the onboarding, execution, interaction of the different LOCUS functionalities used in the different use cases, and the interaction of them with the 3rd party software and external services. A crucial component of the LOCUS platform is the Orchestrator ("OSM" in Figure 2.9), which enables automation in the use of the virtualization infrastructure, allowing the deployment and operation of the LOCUS functions across the edge and core domains.

The following subsections describe the different components of the LOCUS Platform.

2.2.4.1 Virtualized Infrastructure

The LOCUS virtualized infrastructure, as previously mentioned, is based on a hybrid solution, enabling the execution of LOCUS functionalities in both VM and container formats. The following sections describe how this solution is used to enable the execution in both formats.

2.2.4.1.1 *OpenStack*

The OpenStack installation allows to run the UCs' functionalities, virtualized as LOCUS functions, as described in deliverables 4.3 [6] and 4.4 [7], that are implemented as Virtual Network Functions (VNFs), and packaged as virtual machines.

Based on the Queens release [4], it consists of a controller node and three compute nodes. Three different slices have been created to host the different components of the LOCUS Platform. The first one, named “NFVI”, is configured as the virtualized infrastructure to be used for the instantiation of the LOCUS functions. The next one, named “EDGE CLOUD”, is dedicated to the edge domain where the Kubernetes cluster, described in 2.2.4.1.2, is installed. The last one, named “LOCUS-PLATFORM” hosts all the LOCUS Platform components.

OpenStack offers a web Graphical User Interface (GUI), which enables the users of the LOCUS Platform to onboard their VNF images (i.e. a bootable file for executing a VNF) and make them available to the different UCs and PoCs. It also allows access to the different functions, once these have been instantiated through the LOCUS Management and Orchestration tool.

The number of resources dedicated to the OpenStack installation consist of 264 CPUs, 768GB of RAM and ~1TB of HDD.

2.2.4.1.2 *Kubernetes*

The Kubernetes cluster allows to run the UCs’ functionalities, virtualized as LOCUS functions, as described in deliverables 4.3 [6] and 4.4 [7], that are implemented as Containerized Network Functions (CNFs), and thus packaged as docker containers. The cluster consists of three nodes, a master node and two worker nodes.

The Kubernetes cluster is enriched with Rancher [10], a K8s multi-cluster orchestrator that manages the different cluster resources, offering a single web interface. This can be used by the LOCUS Platform users to access their functions running as containers. The resources dedicated to the cluster are the following: 40 CPUs, 80GB of RAM and 600GB of HDD.

2.2.4.2 Virtualization Platform components

While the virtualized infrastructure hosts the LOCUS functionalities, the virtualization platform components are in charge of facilitating their operation.

2.2.4.2.1 *ETSI OSM*

The LOCUS Management and Orchestration module is based on the ETSI Open-Source MANO (OSM) [11] solution, release-ten. The LOCUS functions and the services are packaged following the NFV and ETSI OSM principles and data models, as described in section 3.1.4. When packages are ready, they are onboarded and ready to be instantiated into the virtualization platform.

Taking advantage of the role-based access control in OSM, the three different PoCs are logically isolated by three ETSI OSM Projects, as depicted in Figure 2.10, which allow for support of multi-tenancy.

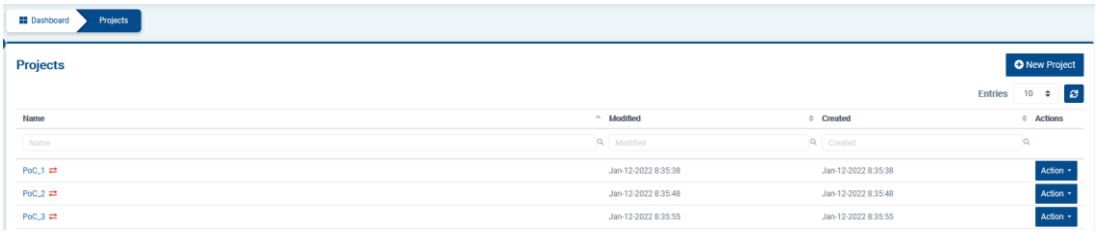

Figure 2.10 OSM projects list

The access to the projects is granted through three different users, each one associated to a single project. Based on this separation, each PoC can independently onboard and run their specific LOCUS functionalities within the LOCUS Virtualization Platform.

2.2.4.2.2 *Container Registry*

The container registry is an application that stores and allows the distribution of the LOCUS functions packaged as Docker containers [12]. The registry is based on the Harbor [13] software, that provides, as in the LOCUS Management and Orchestration tool, a role-based access control, allowing the different PoCs to store the related containers on the specific project related to the PoC itself, as depicted in Figure 2.11.

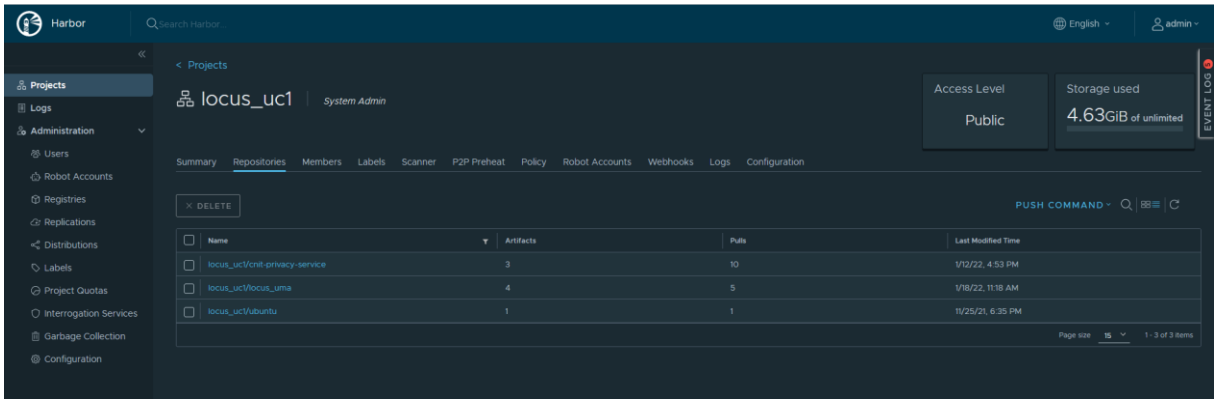

Figure 2.11 LOCUS functions related to PoC 1

The container images stored in this registry are automatically retrieved and downloaded during the instantiation phase of the LOCUS functions from the ETSI OSM orchestrator.

2.2.4.2.3 *Data Distribution*

The data distribution service is based on Rabbitmq [14] software, a message broker that implements Advanced Message Queuing Protocol (AMQP). It allows, through the different topics, to publish and consume the generated data from the LOCUS functions and other LOCUS platform components.

2.2.4.2.4 *DNS*

The DNS service is based on bind9 [15] software. It is used as a complementary tool within the LOCUS Platform, acting as an authoritative DNS for the domain *locus-project.eu*. It is used to easily identify, through the Fully Qualified Domain Name (FQDN), the different LOCUS Platform services, as well as the different LOCUS functions.

The virtual machine where the DNS service is deployed has been equipped with a webservice based on Apache2 software, which provides a Helm Chart repository [16]. This is of extreme relevance for enabling automation in the deployment of the LOCUS functions in the virtualized infrastructure through ETSI OSM, which indeed makes use of Helm to interact with the Kubernetes cluster.

2.3 UMA Testbed Infrastructure

2.3.1 *General Description*

The UMA infrastructure consists of two main groups of resources. On the one hand, there is a cellular network testbed which is fully controlled by the research group. This network can be used to test mechanisms for location-aware management. On the other hand, there are a set of devices that can be used for testing novel location solutions. All of these resources will be described in this section. These resources have been deployed in two classrooms/ labs where all the tests will take place.

2.3.2 *Technologies Overview*

2.3.2.1 Mobile network testbed

The main testbed is an LTE (Long-Term Evolution) network that can be fully managed by the group (including both Radio Access Network - RAN and Core Network, as well as UEs with different capabilities). This network is based on the solution for private networks that Huawei provides, where all the core network elements (Home Subscriber Server - HSS, Mobility Management Entity - MME, Serving Gateway - S-GW, Packet Gateway - P-GW and Policy and Charging Rules Function - PCRF) are grouped into a single compact equipment, namely, the evolved Core Network Solution (eCNS), as shown in Figure 2.12.

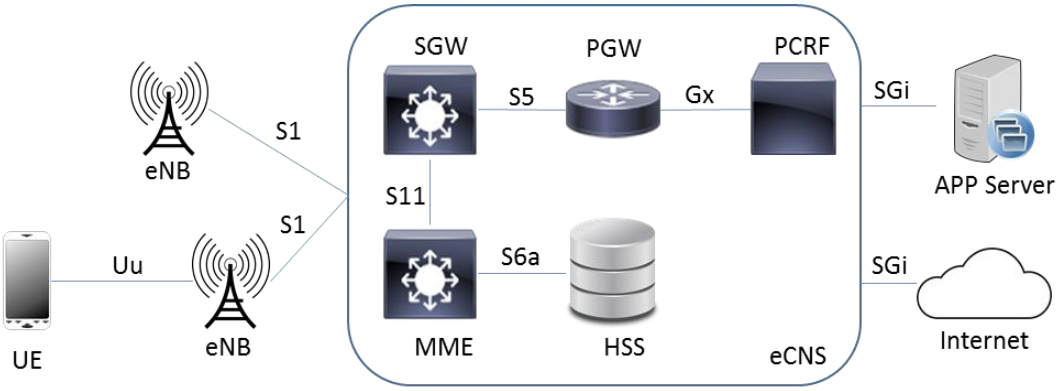

Figure 2.12 General scheme of the LTE network deployed at the University of Málaga

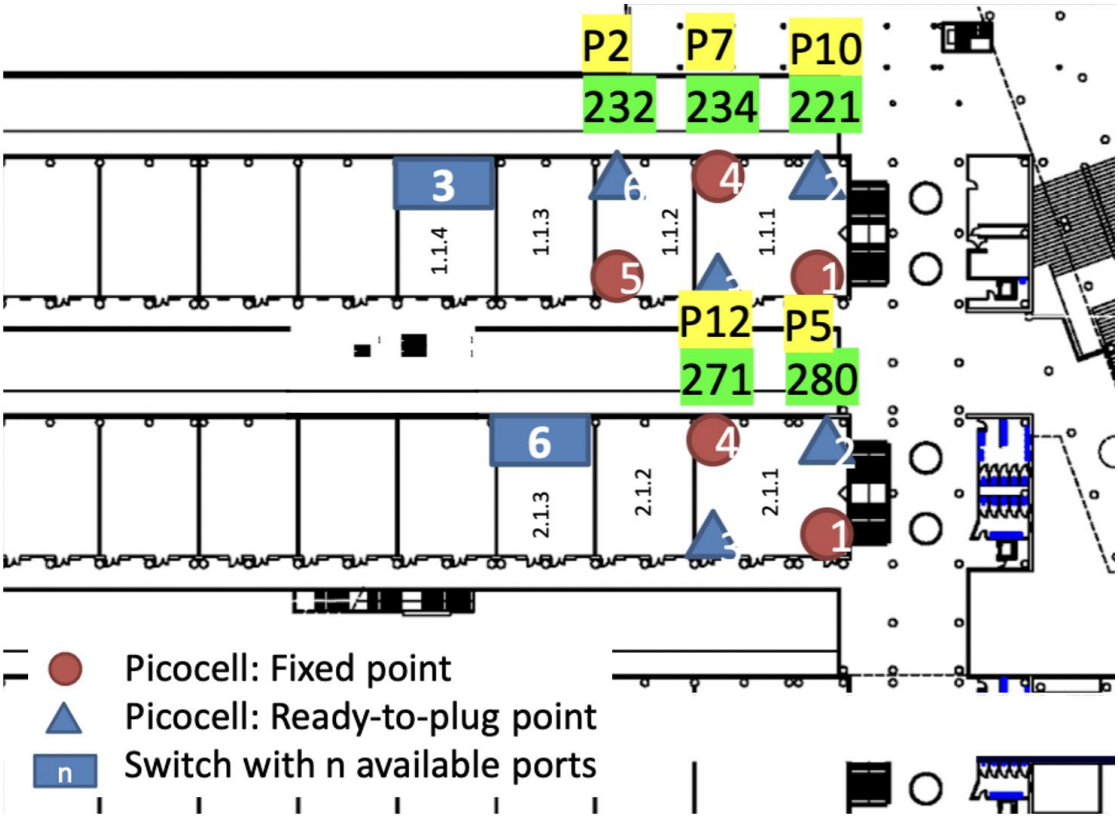

Figure 2.13 Map of the UMA research group's building where the LTE network is deployed

The base stations are 12 BTS3911B picocells, which include a Wi-Fi access point. The infrastructure is completed with the Wi-Fi controller MAG9811, which, together with the iManager U2000 can provide LTE/ Wi-Fi mobility.

Finally, a 24-port GB switch is provided in order to interconnect all the elements, allowing other servers to be added and providing all the equipment access to the Internet.

The location of this network is shown in the map of the UMA research group's building (see Figure 2.13).

The area where the tests will be performed are the labs 1.1.1 and 1.1.2, which can be identified in the previous map. Between the 1.x.x labs and the 2.x.x, there is a corridor and a gap. The corridor is separated from the gap by a metallic grid acting as a handrail.

Additionally, UMA has a 5G network which has recently been deployed but is not yet fully integrated into the LOCUS Platform. The 5G infrastructure is composed by 6 AWHQB indoor base stations and 3 AEQQ outdoor base stations. It consists of a complete 5G Standalone deployment of Nokia commercial equipment. The core network is virtualized into the Compact Mobility Unit (CMU), including the Session Management Function (SMF), User Plane Function (UPF), Access and Mobility Management Function (AMF), Unified Data Management (UDM) and Authentication Server Function (AUSF) components. The 5G Network Architecture is depicted in Figure 2.14. The network backhaul is based on 10G ethernet given the 5G requirements. In order to avoid Internet bottlenecks, UMA has deployed a 10-gigabit infrastructure for hosting their own servers and deploying their applications. Therefore, UEs can connect both to the Internet (1-gigabit) and the UMA-hosted servers (10-gigabit). 15 SA-compatible UEs are available for connecting to this infrastructure, but any compatible device can be connected by providing a SIM card.

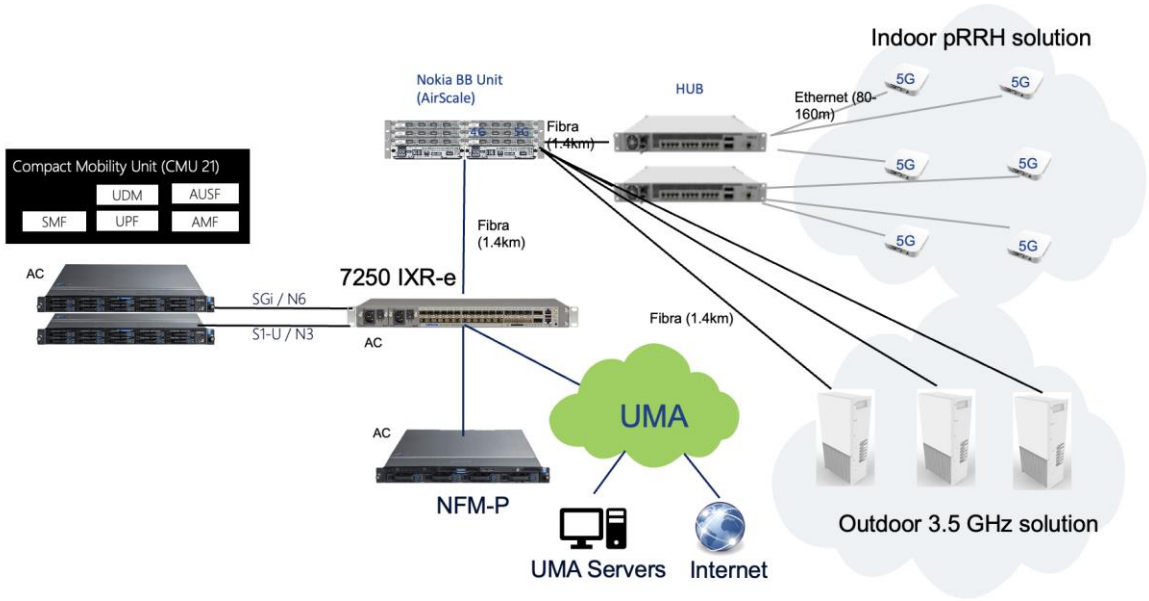

Figure 2.14 UMA 5G network architecture

This infrastructure is private, being the University of Malaga able to access any of the elements either core or radio. Figure 2.15 shows the 5G network elements locations related to the PoC 1 area.

Figure 2.15 UMA 5G network sites

2.3.2.2 Location devices

While UMA uses the LTE network for location, it is complemented by two other location technologies:

- Ultra-Wideband (UWB): a set of 12 Decawave DWM1001 devices, which can act both as anchors (reference points) and tags (location targets) is available and used for precise location. More devices have been ordered to expand the coverage area. One of the UWB devices is configured as a target and linked to a smartphone via a Bluetooth serial port to transmit location information.
- Wi-Fi- Fine Time Measurement (FTM): a set of 6 Google Wi-Fi routers are deployed, providing reference points for devices that support IEEE 802.11mc. A smartphone with this capability is used for testing location with Wi-Fi-FTM.

These technologies are used together with LTE power measurements to provide a precise indoor location using the fusion techniques developed by UMA in WP3 [17][18]. Figure 2.16 shows the location of these devices in the testbed scenario.

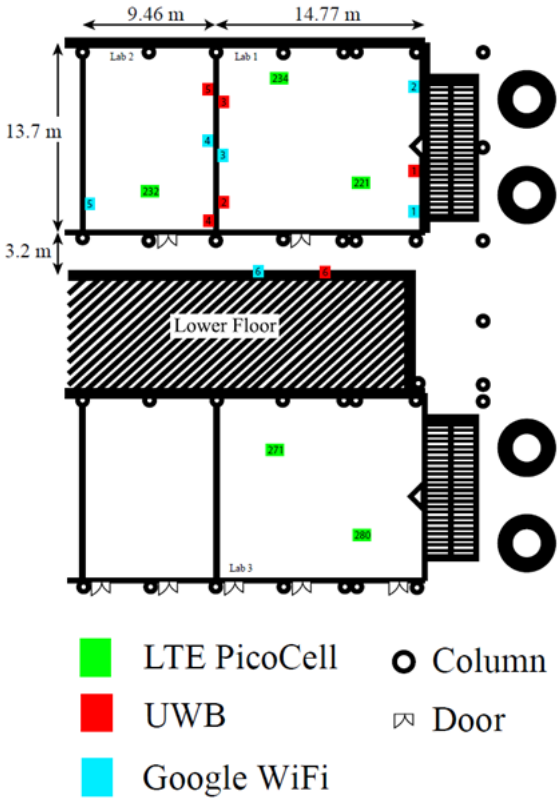

Figure 2.16 UMA testbed scenario with LTE, UWB and Wi-Fi-FTM reference points

2.3.3 *Management Systems and Networking setup*

Together with the eCNS, a tool for the management of the LTE network is provided: the iManager U2000, the same manager used in the public networks deployed by the vendor.

Both the picocells and the eCNS are fully configurable. The picocells can be managed either directly through an LMT (Local Management Terminal) client or by means of the U2000 and eCNS clients. The latter allow the user to fully customize all the network parameters, as well as monitor the network status using built-in and also user-defined KPIs. Furthermore, this PoC allows the user to integrate his own SON applications and algorithms, from Self-Configuration techniques to any Self-Optimizing and Self-Healing application.

The U2000 controller is connected to a Django-based server which acts as a bridge to the LOCUS platform. This bridge server will act as a translator between the messages on the LOCUS messaging broker and the U2000 commands, both for retrieving KPIs and for setting configuration parameters.

2.4 Migration from Local Testbeds / Emulators

As mentioned in the Introduction section, following the roadmap of the WP6 activities, the involved demonstrators have already begun development of their respective components in

an internal development instance. Each respective instance has been designed to follow the initial LOCUS architecture document [19] and later the finalized LOCUS architecture document [1] as closely as possible. The first migration activity that was performed was the installation of the components on the intermediate, "staging" infrastructure of partner NXW which would be built using the same specifications as the OTE infrastructures and would allow the LOCUS partners to start the integration activities before the resources were available, buying precious time in bug fixes on the final infrastructure. In addition to this migration operation, after the review in July 2021, the feedback received from the review committee stressed that partners should move from the simulated/ generated dataset into real LOCUS data that are derived from datasets of network measurement testbeds, e.g. the UMA testbed. This process can be split into two different sections: a) the adaptation of the models and analytics into the data schema that is being produced by the UMA LTE/NR testbed schema, and b) the fine-tuning of each model/service that is included in all PoCs. To be noted that b) does not necessarily guarantee the optimal model results, as -for example- the UMA testbed cannot for example produce data with high vehicular velocity that would be a fitting dataset for existing and new, mobility-based use cases. As a result, a set of demonstration components will continue to use data from the project's accepted simulation engines.

2.4.1 *Incelligent Testbed to OTE Testbed*

This migration operation includes the porting of the Kubernetes [9] configuration files that have been converted from the initial docker-compose YAML files via tools such as kompose [20]. This functionality was already tested in the NXW Kubernetes cluster. For each of the PoC 3 dockerized components (all variants of the LOCUS SDK analytics functions) these configuration descriptors and the raw image files are imported in the analytics service repository component of the LOCUS infrastructure. This will allow simple orchestration commands to instantiate them into the PoC 3 namespace and activate all their interconnections as well as make the external components (Analytics API Gateway) available to the demo FE applications. In addition to the dockerized components, the PoC 3 components that require bare metal installations (i.e. the LOCUS Persistence Module) are being installed manually by the same installation process that was followed also in the NXW testbed. These components include the HDFS [21] distributed file system, Apache Hive [22], Trino.IO [23] and Consul [24] which are placed in a network location that is accessible by the various LOCUS analytics containers via JDBC connection.

2.4.2 *Incelligent Simulator to UMA Network Data*

In order to migrate all the production-ready models of PoC 3 we have started to adapt the measurement-based dataset that was provided by UMA in the input of the system, namely

the data collection function and the fingerprinting model developed in the context of WP3 [25][26][18]. Specialized ingestion scripts are used to convert the datasets into databases that are accessible to the fingerprinting model via the LOCUS SDK. The Apache Hive [22] connector and the schema have been changed accordingly. The performance of this model after extensive fine tuning and changes in the internal structure of its parametrization is shown to provide adequate location prediction results, even after the shift from the simulation schema that was designed by Incelligent for the purposes of fingerprinting. The UMA dataset is a smaller scale dataset that consists of fewer radio access elements, however the accuracy and realism of the measurements as well as the lack of detrimental radio propagation impairments (e.g. shadowing) has allowed for a successful fit of the fingerprinting model which adds to the realism of the demonstration.

In fact, Incelligent's fingerprinting service has already been tested with respect to its adaptation to the UMA dataset. More specifically, the developed model allows for data transformation by ravelling data into data columns per measurement. The different number of sites per location has led to the creation of different datasets with fixed number of sites. In terms of model training, different neural network architectures (feedforward, sequential) with varying width and depth on all datasets have been evaluated. Interestingly, the dataset with only two sites has given the best positioning average error, i.e. ~2m. Furthermore, with respect to error distribution, close to 69% of the predicted positions were within a 2.5m radius from the ground truth positions and approximately 93% within a 5m radius. These results are summarized in Figure 2.17.

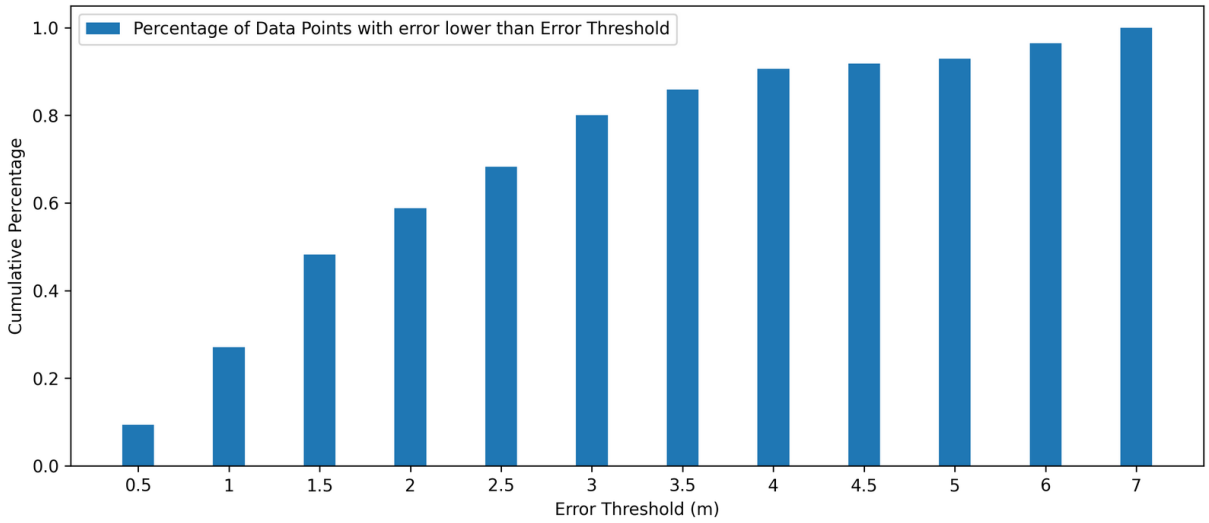

Figure 2.17 Cumulative Percentage (y-axis) of data points with respect to error distance threshold (x-axis)

3 Updates on the existing PoC Use case Implementations

3.1 PoC 1 Updates – Streaming Application

3.1.1 *Service Design*

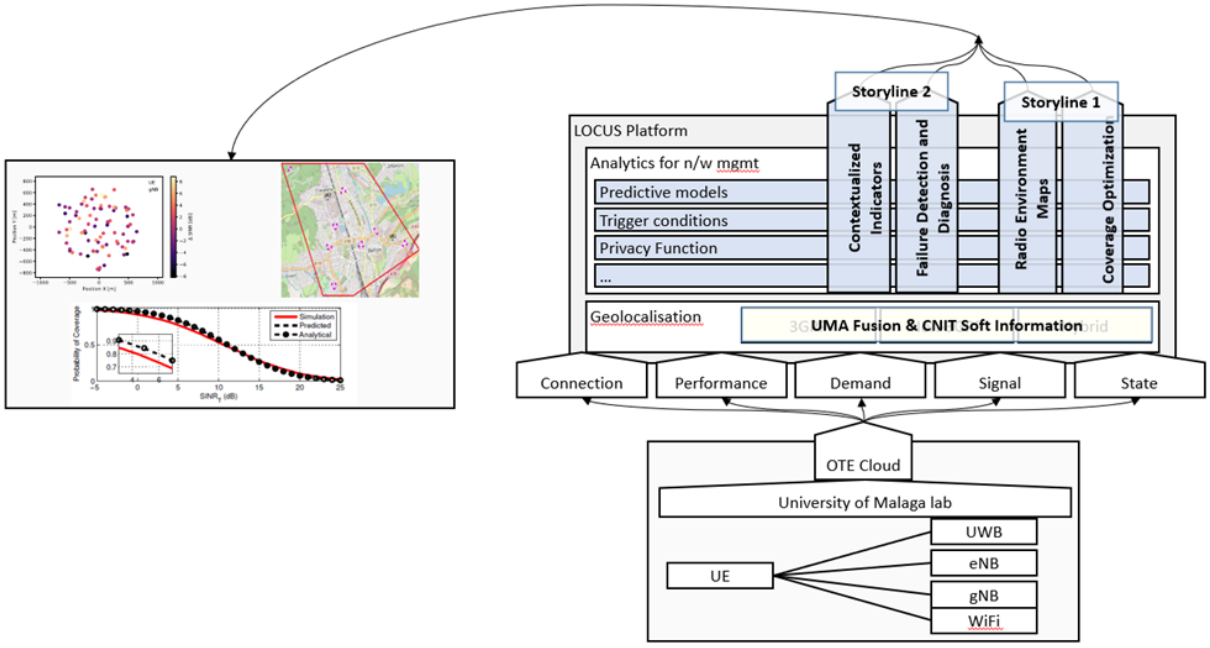

Figure 3.1 PoC 1 overview

PoC 1 consists of a physical infrastructure and a software platform as shown by Figure 3.1.

- The physical infrastructure relies on the integration between the LTE/5G testbed deployed at the university of Malaga's premises, and the cloud servers deployed at OTE's premises in Greece.
- The software platform relies on service pipelines based on virtualized functions, to form two storylines. However, each service has a given function. For example, some services represent a network management function, while others represent a localization function. There are also particular functions used to manage the overall platform, deemed platform functions, such as the privacy function and the OSM MANO used to secure the collected measurements and to orchestrate the different services respectively. Table 3.1 displays the different components of PoC 1 in terms of services:

Table 3.1 PoC 1 components

PoC 1 Components	
PoC 1 Network Management based on Location Information	
Network Management Functions	
Coverage Optimization	Optimizes the coverage based on a radio environment map.
Contextualized Indicators	Fuses of diverse time-dependent data in a selected area and time-frame creating new indicators to be fed to other functions.
Diagnosis and Troubleshooting based on Contextualized Indicators	Used for troubleshooting purposes and failure detection within the network.
Localization Functions	
Fusion Based, Least Squares Estimation	Provides geolocation information based on measurements from different technologies, such as WiFi, UWB, LTE and 5G.
Soft Information	Uses a machine learning algorithm to estimate positions.
Platform Functions	
Privacy Function	Anonymizes the collected geolocation measurements.
OSM MANO	Functions that allow for the orchestration and management of the virtualized functions and the virtual resources.
GUI	Allows for the visualization of data and the services' outputs.

Networks Management Functions: Those functions are used for network management purposes. For example, the Coverage Optimization function is used to optimize the coverage based on a radio environment map which is also based on geolocation inputs for coverage estimation. Similarly, the Diagnosis and Troubleshooting function is used for troubleshooting purposes and failure detection within the network. It also based on contextualized indicators that are fed by geolocation inputs.

Localization Functions: Those functions are used to provide geolocation inputs for the Network Management Functions. The PoC 1 includes two Localization Functions:

- Fusion Based, Least Squares Estimation: This function relies on a set of measurement collection relying on different technologies, such as WiFi, UWB, LTE and 5G. It can be used for two different purposes. Either to provide training dataset to a different Localization Function (i.e.; Soft Information - SI function), or to provide geolocation input for the network management function
- Soft Information: This function relies on a machine learning algorithm to estimate positions that are used as input data for the Network Management Functions

Platform Functions: The Platform functions are used to manage the LOCUS platform. They include, for instance, the privacy function, used to anonymize the collected geolocation measurements. OSM MANO functions allow for the orchestration and management of the virtualized functions and the virtual resources.

GUI: The graphical interface is also considered as a PoC component used to visualize and interactive with the states and outputs of the various services.

3.1.2 *Service View Analysis*

All the different services will be virtualized into containers and pipelined together to build two different storylines. The figures below detail the flow process of the storylines:

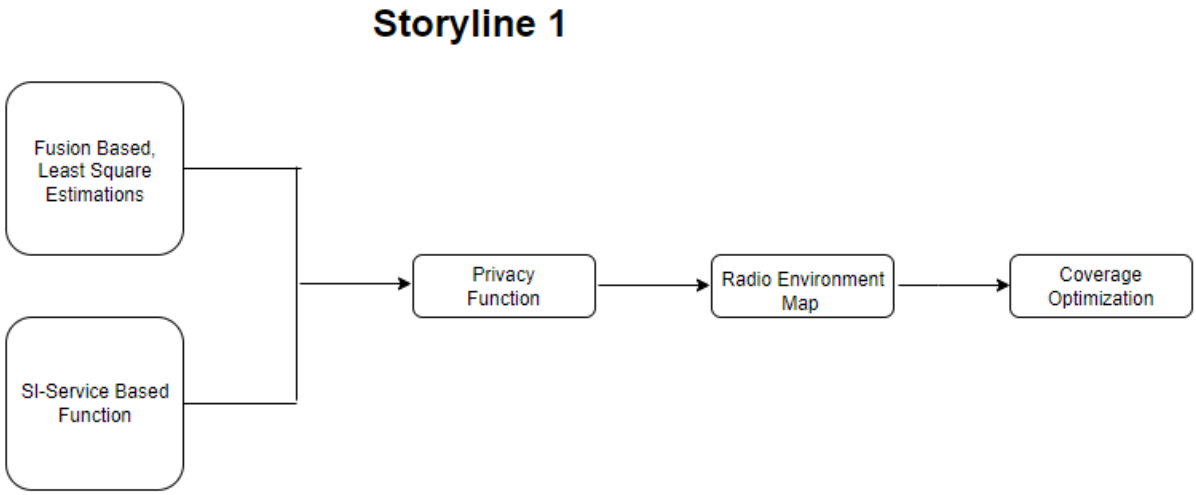

Figure 3.2 PoC 1 Storyline 1

Figure 3.2 displays the different service functions that will be integrated within the LOCUS platform to implement Storyline 1. The localization functions are used to feed the network management functions (i.e.; Radio Environment Map Function and Coverage Optimization Function). However, the outputs of the localization functions pass through a privacy function for anonymization purposes before feeding the network management functions.

Similarly, Figure 3.3 displays the different services used to implement Storyline 2. In Storyline 2, two network management functions will be integrated (i.e.; the Contextualized Indicators Function and the Diagnosis and Troubleshooting Function). Those two functions are also fed by localization functions that generate positions to be anonymized before serving as inputs for network management functions.

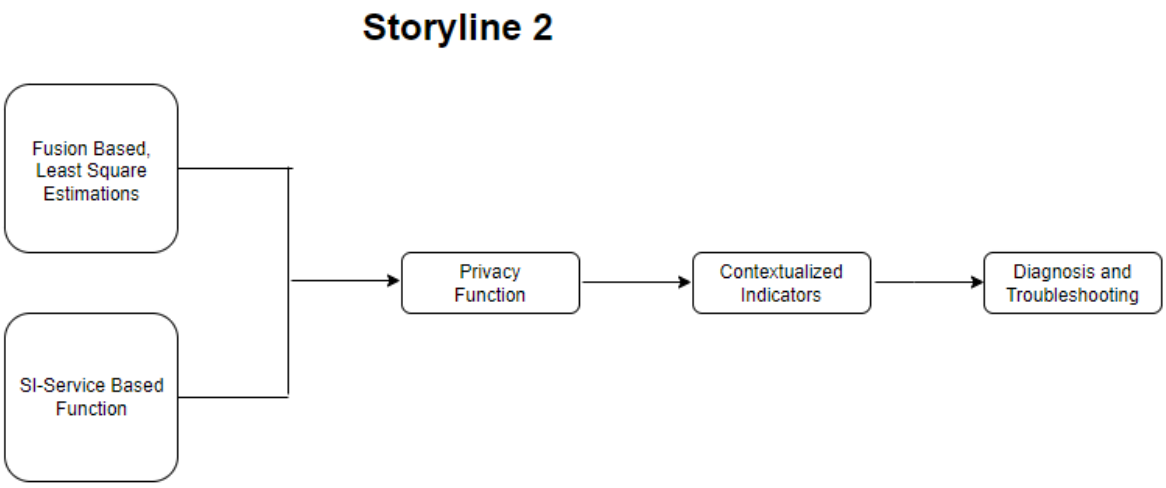

Figure 3.3 PoC 1 Storyline 2

3.1.3 *Demonstratable Scenarios*

To emphasize the geolocation impact on the results, different scenarios based on different geolocation accuracies are to be considered. For example, to properly optimise the coverage, good position accuracies are to be provided for the network management functions. Similarly, to properly detect failures and troubleshoot, accurate contextualized indicators must be built based on accurate positions. The demonstration will mainly focus on the following scenarios:

- Scenario 1: geolocation measurements are provided by only one technology. For example, by only UWB or LTE. This is used to demonstrate which technology provides better position accuracy.
- Scenario 2: different deployments of Access Points are considered. For example, one deployment where a room is not well covered. Another deployment where the room is well covered
- Scenario 3: geolocation measurements are provided by the SI-algorithm.

Using a GUI interface, a user can select the scenario he wants to test and compare the results. By selecting the first scenario, the user is expected to see different metrics evaluating the accuracy of the employed technology used to geolocalize users. For example, the user can compare different curves evaluating the error of Reference Signal Received Power (RSRP) or any different KPI prediction (cf. storyline 2). In this way, the best technology can be evaluated and its impact on the proactive decisions put forward.

The second scenario is also used to provide metrics that put forward the importance of consistent data collection measurements. For example, when a room is not well covered, measurements would be inadequate, and thus the impact on the quality of analytics or proactive decisions would be severe. However, when the room is well covered and we have enough measurements, estimations are accurate and better decisions can be made. A user

can see the different errors and the different troubleshooting decisions and diagnosis resulting from the different scenarios of deployment and compare their performance.

The third scenario is used to evaluate a potentially interesting localization algorithm. Errors and decisions that are based on this algorithm are expected to be tested in order to evaluate its accuracy.

With respect to the coverage optimization scenario, its objective is to assess the added value of geolocated measurements for network coverage optimization. In this scenario, the following functions are involved in the main storyline:

- Radio Environment Map: this function takes as input LTE RSRP measurements associated to measurement locations and gives as output an RSRP map for each LTE base station.
- Coverage Optimization: this function takes as input the RSRP maps provided by the Radio Environment Map function and analyses the coverage on the area of interest. If a coverage degradation is detected, then the Coverage Optimization function determines a new parameter setting that guarantees the required coverage level. In the current scenario, as the parameter to be tuned is the transmitted pilot power, the Coverage Optimization function is able to find the optimal setting based on the knowledge of the different base station coverage maps.

In addition to this main storyline, and in order to show the impact on the performance of the coverage maps accuracy, two additional storylines can be show-cased:

- In the first case, different levels of geolocation accuracies, related to location techniques performance, can be tested. The accuracy of the coverage maps will be impacted, and we aim at showing the impact of having high or low accuracy on the performance of the Coverage Optimization algorithm. In other words, we expect to solve more efficiently the coverage issue if we have more accurate location measurements.
- In the second case, we can vary the number and distribution of RSRP measurements. With low measurements density, the coverage map accuracy will be degraded, and we can also show the impact of the quality of the coverage maps on the performance of the Coverage Optimization algorithm.

In terms of visualization, the Coverage Optimization scenario requires to show: the measurement locations on a map; the coverage maps before and after that the coverage Optimization algorithm changes the settings. We optionally need to show where we are in the process, and to give some textual information (e.g. location technique and associated error estimation, coverage rate, coverage map accuracy, etc.).

3.1.4 *Initial Integration into the LOCUS Virtualization Platform*

The integration of the LOCUS functions into the LOCUS Virtualization Platform initially started at Nextworks's Lab, is continued at the OTE infrastructure.

As of today, the LOCUS Platform is able to host two different scenarios related to the PoC 1 and fully integrated to the LOCUS Platform. The first service is related to the precision accuracy, and the second one is related to the privacy service.

As described in Deliverable D4.3 [6], the integration process of functions and services into the LOCUS Platform starts with each function and service implementer/ provider providing a set of requirements information, through the Function Template described in D4.3 ([6], Annex A) This document is used to: i) build an initial architecture of the virtualized service, ii) separate the different functionalities of the service, iii) identify the interconnection among the functionalities, iv) identify input/ output datatypes, v) identify the required LOCUS Platform components.

The completion of the above process allows the LOCUS functions and service providers involved in the virtualization process to derive the design for their virtualized services. After this design phase, each function involved in the virtualized service is packaged into a docker image, through a containerization process, properly adapted to allow configuration of custom parameters required to correctly run the service. These images are stored in a LOCUS Platform component, the LOCUS Container Registry, described in 2.2.4.2.2, and are ready to be used within the LOCUS Platform.

After the packaging of the software in docker images, another packaging is performed. The functions and the services need to be packaged to be handled by the LOCUS Management and Orchestration tool.

This process consists of two different stages: i) virtualization of the LOCUS functions, and ii) composition of the LOCUS service associated with the LOCUS functions.

The first stage consists in the creation of a Kubernetes network function (KNF) for each of the functions. ETSI OSM, the LOCUS Management and Orchestration component, allows to package the functions using two different technologies: i) helm chart, or ii) juju charms.

Helm Chart [16] is a Kubernetes standard deployment, and its base element is the chart. It consists of one or more template files in a YAML format that describe the function and its deployment procedure. Instead, Juju Charm [27] base element is the Charm, a structured bundle of YAML descriptors and script files based on Python that allows the instantiation and the configuration of the LOCUS function described in it. In the context of the LOCUS platform, both solutions have been used, even if the juju charm solution is to be preferred, since it allows dynamic automated configuration through the use of day-1/day-2 configurations with configuration primitives expressed and parametrized in the VNF Descriptor.

After the creation of the juju charm, or the helm chart, the process of packaging proceeds with the creation of the function descriptor, namely VNFD (Virtualized Network Function Descriptor), which describes the function requirements in terms of virtualized resources for its deployment and execution based on ETSI NFV SOL006 [28]. The main components of the VNFD, as depicted in Figure 3.4 are: 1) the reference to Kubernetes cluster networking whereby the function will be exposed to the other functions and platform components, and 2) the juju bundle referred to the function itself.

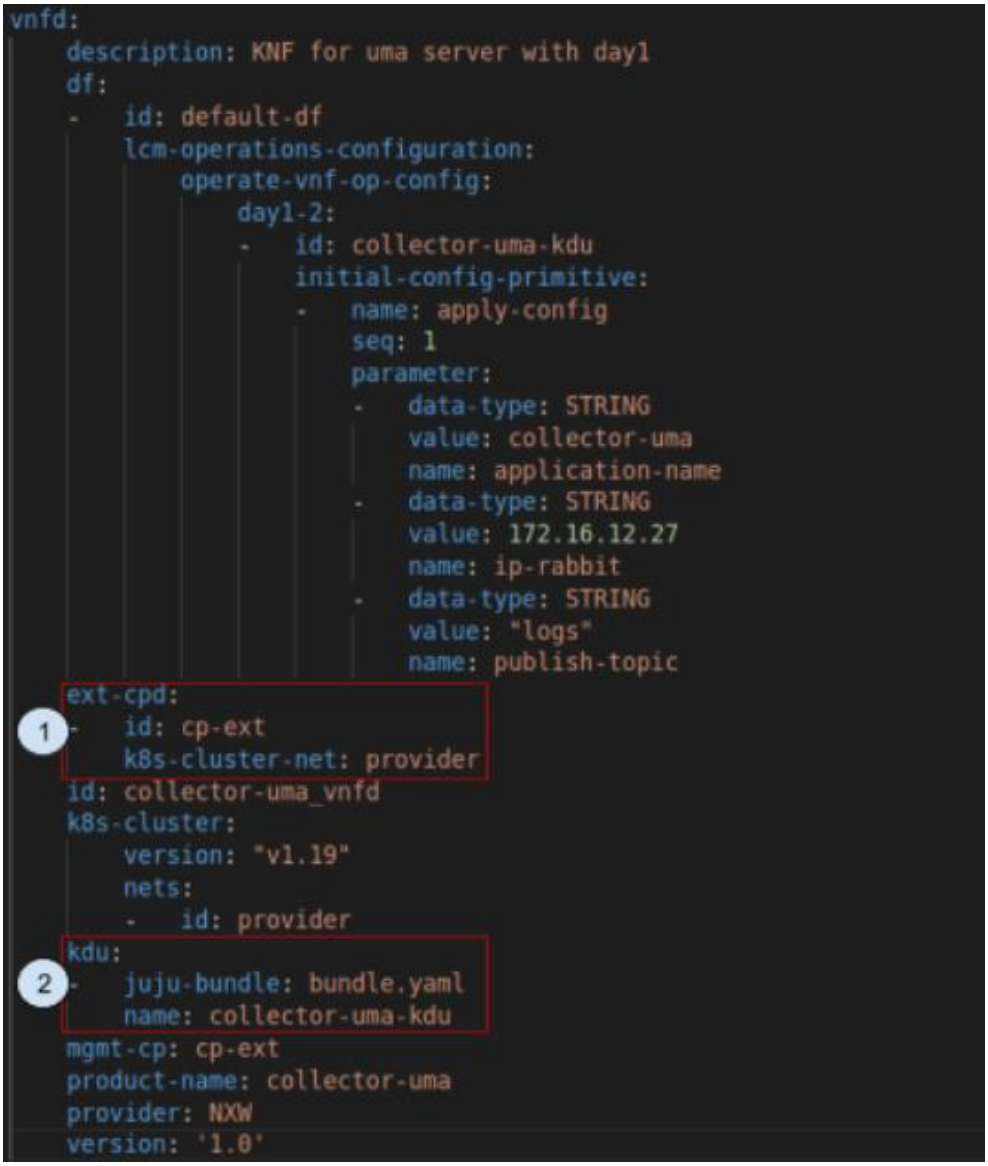

Figure 3.4 VNF Descriptor for LOCUS Collection function

The last package performed before making a LOCUS service fully deployable, configurable and manageable over the LOCUS platform is the creation of the LOCUS Service, which is modelled as Network Service Descriptor (NSD), again based on ETSI NFV SOL006 [28]. When this last

element has been completed, the descriptors need to be onboarded in the LOCUS MANO platform, enabling the LOCUS platform users to deploy and run their functions within the platform.

The following subsections will describe the high-level architecture of the fully virtualized LOCUS services into the LOCUS Virtualized Platform.

3.1.4.1 High Precision Accuracy Service

The following figure (Figure 3.5) depicts the architecture of the high precision accuracy service.

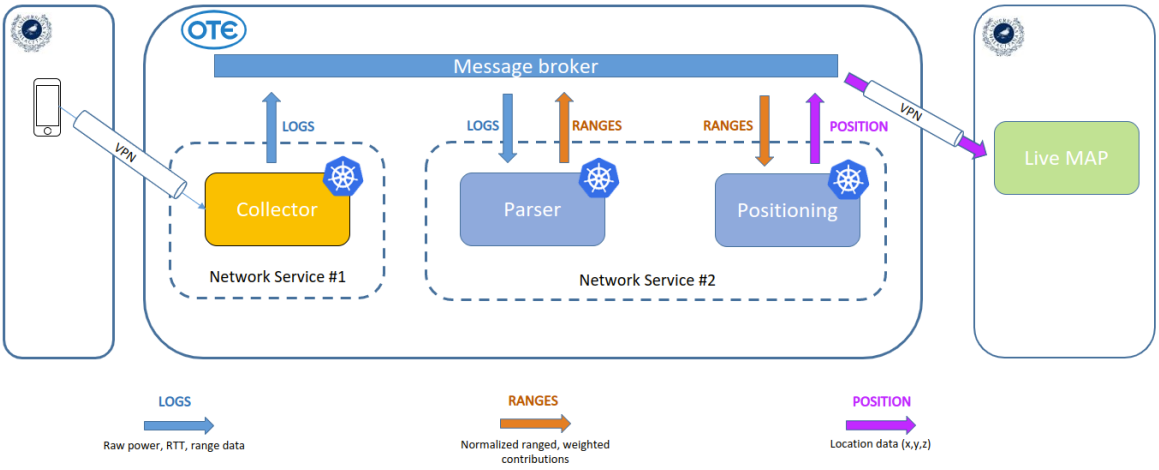

Figure 3.5 High precision accuracy service

The service is composed of two different NFV Network Services. The first one, the collection service, allows the UEs connected to the UMA infrastructure to send their location data to the LOCUS Platform. The second service, namely processing service, is composed of two different functions, in charge of cleaning the data sent by the UE and calculating the positioning of the user. The data is sent back to the LOCUS Platform, using the data distribution movement component described in 2.2.4.2.3, and can be read by other LOCUS functions and services functionalities. In this scenario, the consumer of high precision accuracy virtualized service is a Graphical User Interface software showing the live map of the UMA laboratories.

3.1.4.2 Privacy Service

Figure 3.6 depicts the architecture of the privacy service and the interaction with the generic platform modules. The service is composed of a single NFV Network Service that contains two functional blocks. The first one, the *k*-anonymity and aggregation function receives users' requests, processes the data to make the output related to one user indistinguishable from at least other *k*-1 individuals and uses aggregate responses to respond to 3rd party; then, the

aggregate request is pushed to Localization Base Service (LBS). The second block, the storage data volume, maintains the data structure used from the first function, which is a subset of data provided by the data persistent module.

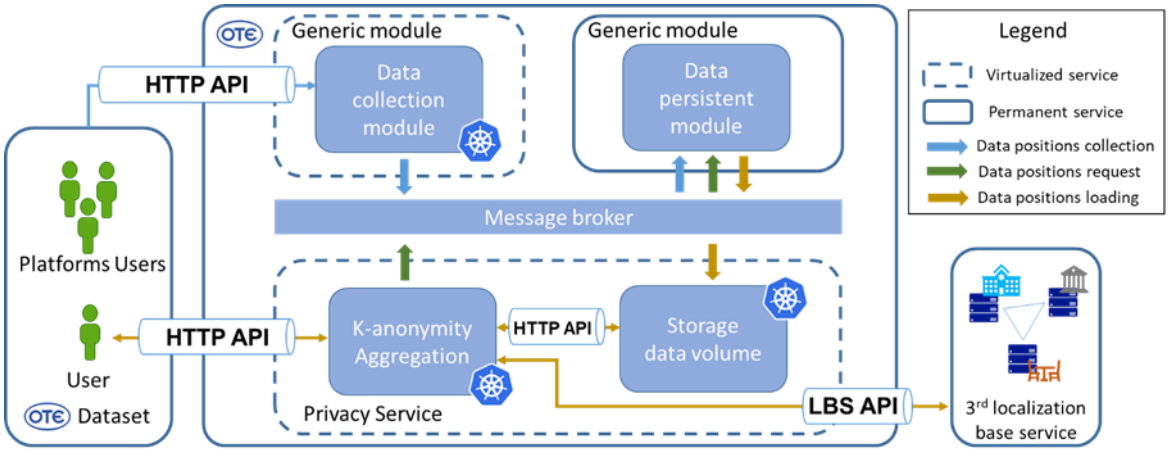

Figure 3.6 LOCUS privacy service

3.1.5 *Virtualization of SI-based localization*

Localization algorithms based on the concept of SI have been developed and designed in the context of the WP3 work on localization enabler for 5th generation (5G) networks [29]. In the deliverables D3.1 [25], D3.2 [26], D3.3 [17], and D3.4 [18], the performance improvements in terms of localization accuracy compared to classical methods have been evaluated in 3rd Generation Partnership Project (3GPP) standardized scenarios, using both radio access technology (RAT)-dependent and RAT-independent measurements.

As part of the work in WP6, SI-based localization will be integrated in the LOCUS platform and will provide accurate location information of multiple user equipment (UEs) connected to the network. This information can be exploited to extract analytics for the different use cases (UCs) defined by the LOCUS project for PoC 1. In order to integrate SI-based localization algorithms in the LOCUS platform, the code used to obtain the results reported in WP3 deliverables has been rewritten using an open-source programming language and reorganized to allow the deployment in the platform. In particular, SI-based localization has been split in three different virtual network functions, namely *Collector*, *Parser*, and *Positioning*, which will compose the network service. In order to exploit the measurements provided by UMA infrastructure, the proposed network service shares a similar structure with the one described in Section 3.1.4.1. However, the virtual network function used to provide position estimates implements a different localization algorithm developed according to the SI approach. The following design choices have been made:

- each of the functionalities is developed for being implemented on standalone Docker [12] containers running the operating system Ubuntu 20.04 [3]; and

- each container has one exposed port for communicating through an AMQP protocol using the RabbitMQ [14] message broker (and possibly Kafka [30]). In particular, three different topic exchanges are used to ensure the connection between the docker images. Each package transferred through the RabbitMQ bus contains all the measurements needed to accomplish SI-based localization.

The tasks accomplished by the three virtual network functions are described in the following text:

- *Collector:* Collects localization measurements data provided by the devices connected to the network. The data are forwarded to the logs exchange encoded in JSON format.
- *Parser:* Pre-processes the data to make them fully compatible with SI-based algorithms, such as conversion of time/ power measurements to distance measurements.
- *Positioning:* Infers the target position using the developed SI-based approach exploiting the information contained in the packages sent through the ranges exchange. It exploits generative models pre-loaded in the Docker container and trained during an offline phase. The estimated positions are forwarded to the positions exchange.

As initial testing, the proposed architecture for SI-based localization was tested on a single machine with the docker images connected through a RabbitMQ bus running on localhost using the same dataset used to obtain the results reported in WP3 deliverables (see Figure 3.7). The deployment and configuration in the LOCUS Platform of SI-based localization is currently being finalized. Integration with the LOCUS platform will reflect the same functional split and will use as input measurements gathered from real devices. Lastly, the generative models used by SI-based localization will be obtained using machine learning (ML) pipelines via the LOCUS Platform as described in deliverables D5.1 [31], D5.2 [32], and D5.3 [33]. This will allow flexibility in terms of supported measurements and refinement of already trained model using new data.

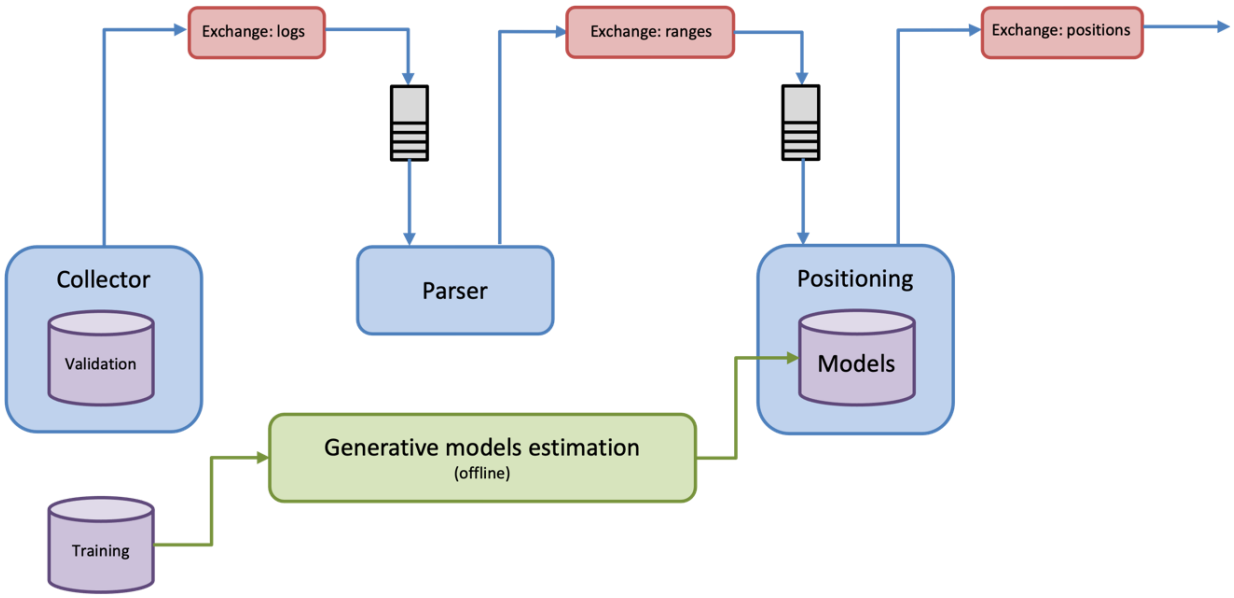

Figure 3.7 Functional components used to test SI-based localization, where the Collector uses a simulated dataset.

3.2 PoC 2 Updates

3.2.1 *Data Flows in TEI Simulator*

The simulation of the behaviour of the 5G positioning in a real context is made by starting from a set of input data describing the spatial distribution of the position estimation accuracy. For each position, the estimated average value and the standard deviation is provided.

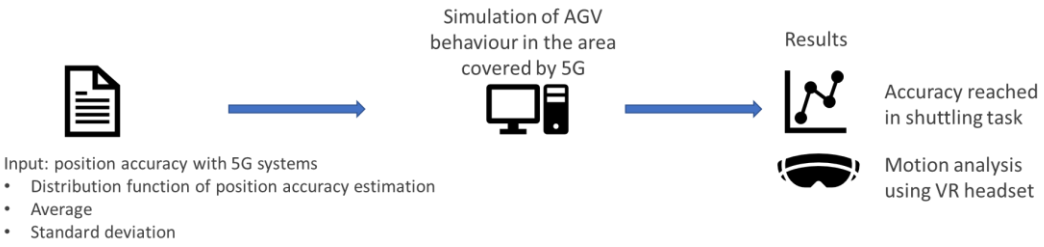

Figure 3.8 Main data flow for PoC 2

As shown in Figure 3.8, these input data are provided to the simulator that simulates the behaviour of an AGV shuttling freights in a port area. The simulation allows the evaluation of the immediate and of the cumulative position errors showing the consequences on the achievement of the shuttling tasks. The simulated AGV is controlled by a fuzzy nonlinear controller that can compensate the cumulative errors.

The simulator architecture is shown in Figure 3.9.

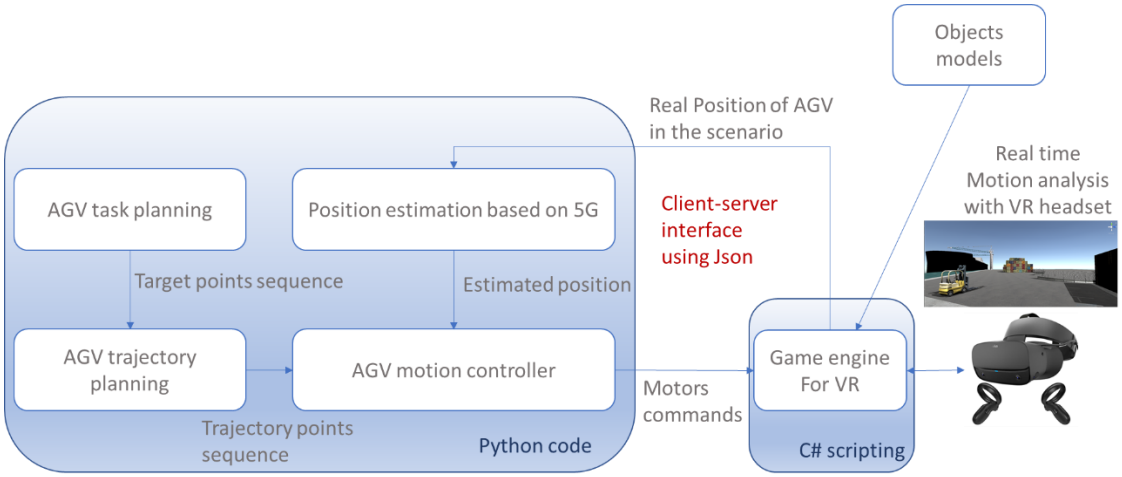

Figure 3.9 Simulator architecture

The simulator includes a set of functions. The task planner determines the target points of the task. For instance, if the vehicle is in a rest location and must reach a position in the warehouse to pick a freight and then transfer it to the crane for loading, the task planner determines the location of the freight to pick and the position where the object should be delivered. It can also introduce other target points in between to constrain the trajectory that the AGV will follow.

The set of points provided by the task planner is used by the trajectory planner to calculate the detailed trajectory the vehicle should follow. Apart from the constraints provided by the task planner, it makes use of a map of the working area and the position of known objects in it. The trajectory is usually calculated using the A* algorithm, first described in [34]. Once the trajectory is calculated, this is passed to the motion controller that controls the movement of the vehicle along the trajectory. In the simulator a non-linear fuzzy controller is used as the AGV controller. The advantage of a non-linear controller on a PID (Proportional Integral Derivative) is that it can have a faster reaction time keeping a high accuracy in the movement control.

The obtained motor commands are then transferred to the game engine managing the VR environment. The VR part updates the position in the environment and determines the position of the AGV in the environment. This position information is fed to the 5G position estimation function that estimates the position of the AGV and provides it to the AGV motion controller. This is the only positional information got by the motion controller to determine the next move based on the trajectory it should follow.

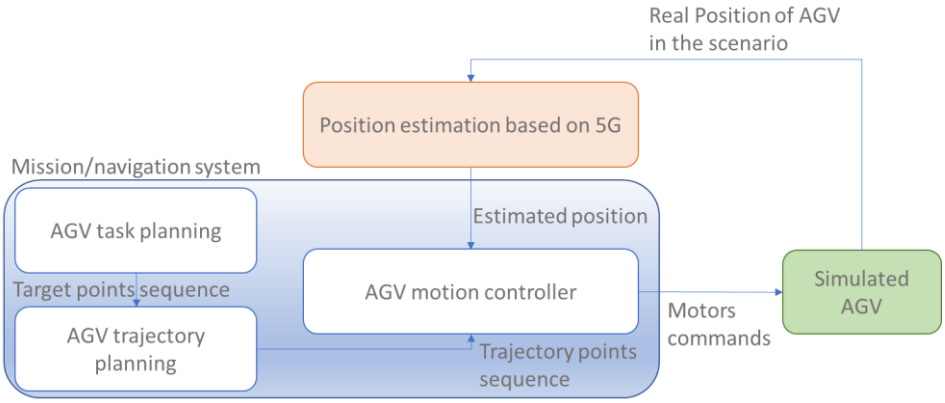

Figure 3.10 Main data flows in the simulator

The data flows exchanged in the simulator are the following:

- Target points sequence
- Trajectory points sequence
- Motor commands
- Real position of the AGV
- Estimated position of the AGV

The **Target points sequence** is a list of (x, y) cartesian coordinates taken with respect to the main Coordinates Reference System used in the VR environment. Each point represents a position the AGV must reach.

The **Trajectory points sequence** is a list of (x, y, t) cartesian coordinates taken with respect to the main Coordinates Reference System used in the VR environment plus a tolerance. Each point represents a position the AGV must reach with a specific degree of tolerance given in relative percentage.

The **Motor commands** is represented by a vector in polar coordinates with respect to the AGV's coordinates reference system, whose magnitude gives the amplitude of the next movement step and the angle between the vector and the axis of the AGV the direction to follow.

The **Real position of the AGV** is a couple of (x, y) cartesian coordinates that provides the absolute position of the vehicle in the VR environment with respect to the main coordinates Reference System. The map used in the 5G position estimation function uses the same coordinates system of the VR environment, so that a 1:1 mapping of positions is guaranteed.

The **Estimated position of the AGV** is a couple of (x, y) cartesian coordinates that provides the estimated absolute position of the vehicle in the VR environment respect to the main coordinates Reference System. This couple is used by the AGV controller to determine the next move.

3.2.2 *Investigation for onboarding to LOCUS Demonstration Platform*

The vehicular mobility environment simulator that is part of the PoC has been initially an external entity to the LOCUS Demonstration Platform. However, partners are considering the possible integration in the demonstration platform as a component that will be performing what-if analysis scenarios. This solution, as well as other alternatives, will be further investigated in the following period to find the best way to achieve said integration.

The concept involves this system to be onboarded to the LOCUS demo platform using the LOCUS SDK framework (as a containerized internal LOCUS analytics function) and it will use as input the location information generated by the fingerprinting models, i.e. models for positioning developed in the context of WP3 and stored in the LOCUS Persistence Module. It will then require user configuration in order to (a) replay the location history into its virtual mobility environment and (b) apply mobility variations that will generate hypothetical new locations of the mobile terminals (including vehicles). These can be assumed operations of the same category as the trajectory prediction module (WP5) which generate new location information based on predictive or hypothetical data. These outcomes can then be further processed by other LOCUS Analytics functions by being stored in a separate "generated location schema" of the LOCUS Persistence Module.

3.3 PoC 3 Updates – Core Infrastructure

The updates that have been conducted on existing components of PoC 3 include richer visual information on the user of the geospatial analysis dashboards and improved real-time capabilities for: (a) Transportation profile analysis, (b) Flow monitoring (i.e. Customer retail behavioural profile analysis in WP4), and (c) Crowd mobility analytics. These changes can be shown in subsections 3.3.4, 3.3.3 and 3.3.5 respectively. In addition, a set of platform-related updates have been conducted to allow for the deployment of the services in the final demonstration platform, including a) dynamic dockerized spawning/ de-spawning operator and b) service name and port reverse lookup service via the Consul.

3.3.1 *Service Discovery Module Implementation (Consul)*

During the deployments at the intermediate testbed in the NXW lab, it was discovered that there was an issue with the static IP / port configuration and the dynamic spawning of containers that occur when Kubernetes is in charge of the network stack (as opposed to the pure docker which was used in the on-premises installations). Therefore, the system needed to be extended to include a component that would store all the network locations centrally and would map it to a specific service name (e.g. the LOCUS fingerprinting service container) in order to allow for inter-communication between the related LOCUS analytics functions. The Consul system was also added in the descriptor of PoC 3 and is now implementing the

functions of the “Service Discovery function” component that was also described in D2.5 [1]. This made the static configuration of IP/ports obsolete and allows for greater flexibility for dynamic shutdown and spawning of the LOCUS SDK containers.

3.3.2 *LOCUS SDK Spawning / de-spawning Function Implementation (docker)*

Part of the LOCUS internal architecture components is the Data Operations Component, capable of interfacing with the virtualization layer (Kubernetes/OSM) to define the required LOCUS SDK function instances. These need to be generated in time for each step of the various LOCUS pipelines, including the pipelines of PoC 3. This interface (otherwise referred to as the OSM NB API) has been implemented for demonstration purposes using local docker commands that perform the spawning, function invocation and de-spawning process dynamically. In this sense, the resource utilization of the container remains to the minimum at each execution, freeing the CPU/ memory and disk resources in every step and maximizing efficiency while keeping the container base clean for easier monitoring and housekeeping.

3.3.3 *PoC 3 Flow Monitoring UI Updates*

With respect to the Flow monitoring use case, the user interface has been improved in terms of enriching the type and quality of information displayed in the front-end.

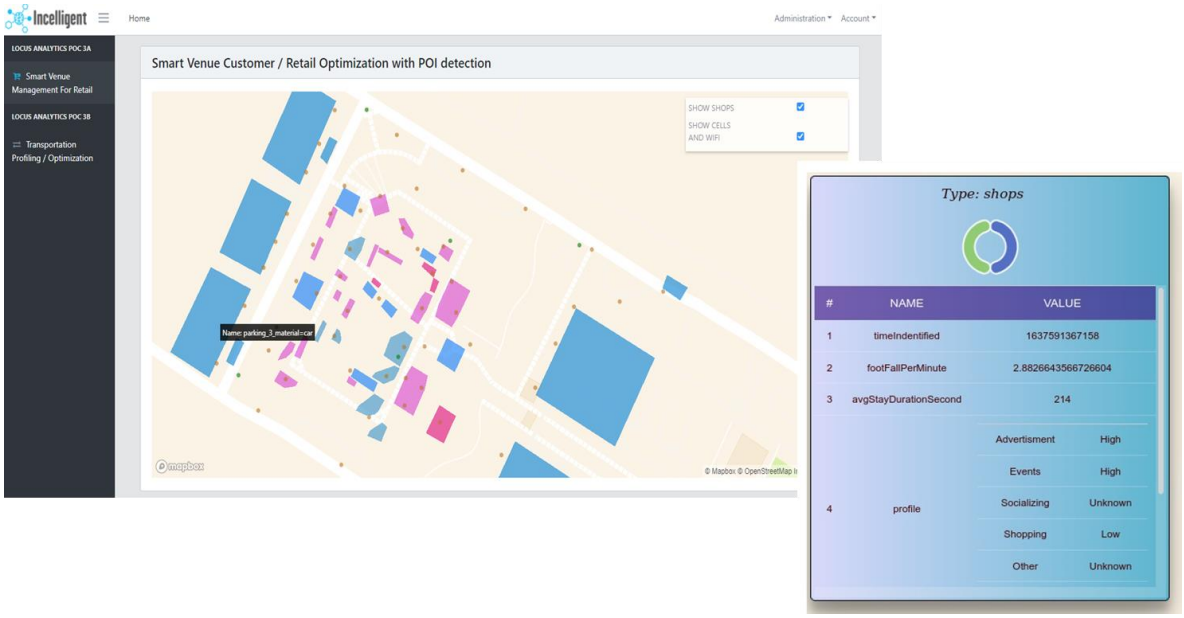

Figure 3.11 UI updates for the Flow Monitoring use case

More specifically, in Figure 3.11 (left side), the overview of the area displaying shops as well as cells and WiFi access points is presented. The map can be configured by the user to present the shops only and/or the cells and WiFi access points. Furthermore, additional POI area metrics/ information can now be displayed through a pop-up window (Figure 3.11, right side),

i.e. time of detection, footfall per minute, average stay of duration, commercial profile characterization.

3.3.4 *PoC 3 Transportation UI Updates*

Similarly to the above, for the Transportation use case, several extensions related to the provision of velocity analytics have been performed.

User can visualize all paths in the area under question through a simple tick in the map widget. In the same way, shops and/or cells can be visualized on the map. Furthermore, all paths in order of popularity/ importance can be presented in a pop-up window (see Figure 3.12).

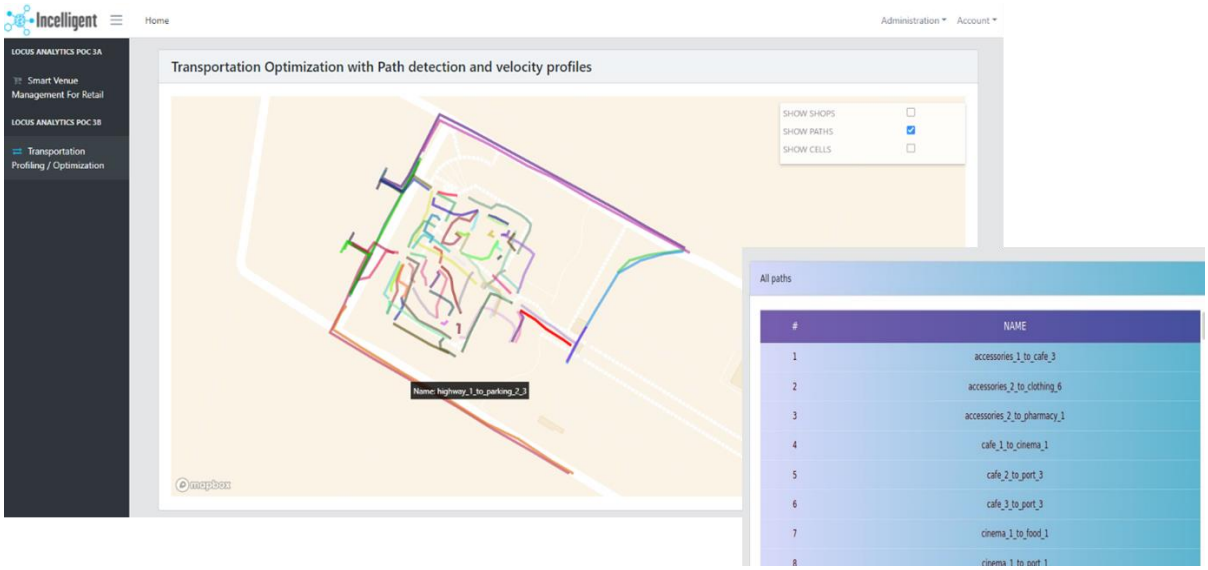

Figure 3.12 UI updates for the Transportation use case

3.3.5 *PoC 3 Crowd Mobility Analytics UI Updates*

The crowd mobility analytics has an initial web interface that includes a dashboard of group mobility insights (group inference) where metrics such as number of people in the vicinity, number of groups, and the number of people in each of the detected groups are displayed. The dashboard was previously offline and not dynamically updated due to time constraints. The current version updates the metric values dynamically and more frequently. The analytics work on the offline dataset that is on CouchDB [35] and the dataset is labelled with ground-truth values from the controlled testing scenarios. The integration of the group mobility insights into the PoC 3 core infrastructure is considered to share the analytics results of the group inference by an HTTP REST API.

4 New PoC Use Cases Design and Implementation

4.1 PoC 3 - Collision Detection for Self-driving Cars

The purpose of this use case is to provide a new service on top of the LOCUS Platform that will utilize location information within the boundaries of a predefined large-scale area in order to provide a safety/ hazard detection framework for self-driving cars. The direct consumers of this service are either the smart-city/ smart-transportation providers or directly the manufacturers of the vehicles (cars, buses, etc.) that will feed this information into their already existing self-driving navigation system to provided added safety. The system will use internally re-trainable mobile trajectory prediction models, developed within the scope of LOCUS [36]. The latter will generate, in real time, possible trajectories of the detected vehicles within the boundaries of the venue. In addition, the usage of preconfigured geometries (i.e. shapes that have been stored in the LOCUS shape persistence schema that describe the topology of the venue) are then correlated using the LOCUS geospatial correlation module (also used in other PoC 3 components) to identify the projected trajectories probability of collision. Overlapping trajectories are used in combination with the trajectory confidence (model outcome) in order to generate real time events/ messages that inform the API users that there is an impeding collision. For the purposes of this demonstration, the application will expose the necessary services and will also have an indicative dashboard that will show the information generated by this service as it would be fed into an external real-time system. It must be stressed that the time-criticality of this application forbids it to be used as a real time dashboard, and this is done only for demonstration purposes.

4.1.1 *Service Design*

This PoC consists of one primary service and two secondary services. The primary service will start the operation of the algorithm and the two secondary ones will allow for the initial configuration of the resources as well as the monitoring of the results in a dashboard.

Table 4.1 Vehicle trajectory prediction / collision detection services

Service Name	Inputs	Process	Outputs
Venue Shape meta-data configuration (secondary)	List of shapes that are obstacles within a venue and the definition of the venue.	The shapes (GeoJSON[1]) that are supplied by the REST API will populate the LOCUS Persistence Module and will then be accessible to the service configuration input definition for the execution of the main service.	Confirmation of the successful ingestion of the shapes (syntax-checking and IDs of the insertion).
Venue Trajectory Predictive Analysis Initialization (primary)	Definition of selected venue, obstacles and detected mobiles that will be included in the trajectory detection algorithm.	The system will begin collection of the real-time generated location information provided by the fingerprinting topic and feeding them into the sequential trajectory prediction module. This will generate a new message type, with the projected locations of the mobiles. These locations are combined with the actual locations and consumed by the shape correlation module using as input the selected shapes in order to generate the collision events. These collision events are then produced as a topic which is then fed to the backend analytics system for visualization.	Various intermediate Kafka topics related to its operation and finally the message flow of collision events, including their trajectories, model certainty, impact certainty, and severity based on distance from the obstacle. It also returns a "monitoring ID" to filter the collision events in the front end.
Venue Trajectory predictive analysis dashboard meta-data (secondary)	The model execution monitoring id to use it as filtering for the collision events.	This is a service to provision all the render-able/ displayable information in the trajectory prediction dashboard (all the serializable information provided in real-time by the collision topic).	The serialized collision topic information for the selected monitoring execution ID.

[1] ArcGIS Online website, https://doc.arcgis.com/en/arcgis-online/reference/geojson.htm

The operation of the service is based on the design that was performed in D2.5 additional PoC scenarios [1], but in a tuned-down version that is only created for demonstration purposes as it is visible in the schematic (Figure 4.1).

We can see that the main operational pipeline is maintained, however the real integration with the production self-driving cars backend and the communication with actual handheld devices (as receivers) has been tuned to simple generation of a dashboard UI that will be analysed in the service view analysis.

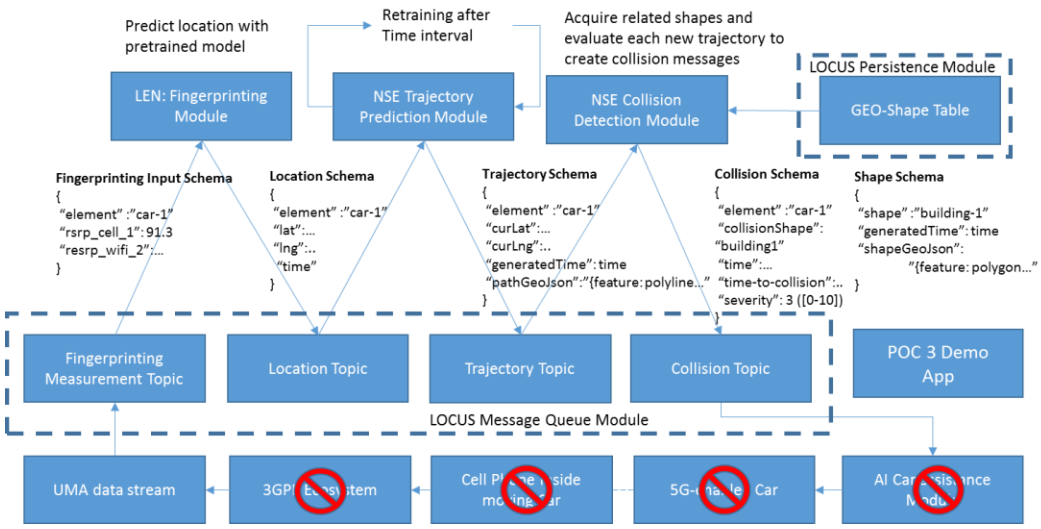

Figure 4.1 Trajectory-assisted self-driving cars service realized flow

4.1.2 *Service View Analysis*

In this section, the indicative front end component decision of the trajectory prediction is presented.

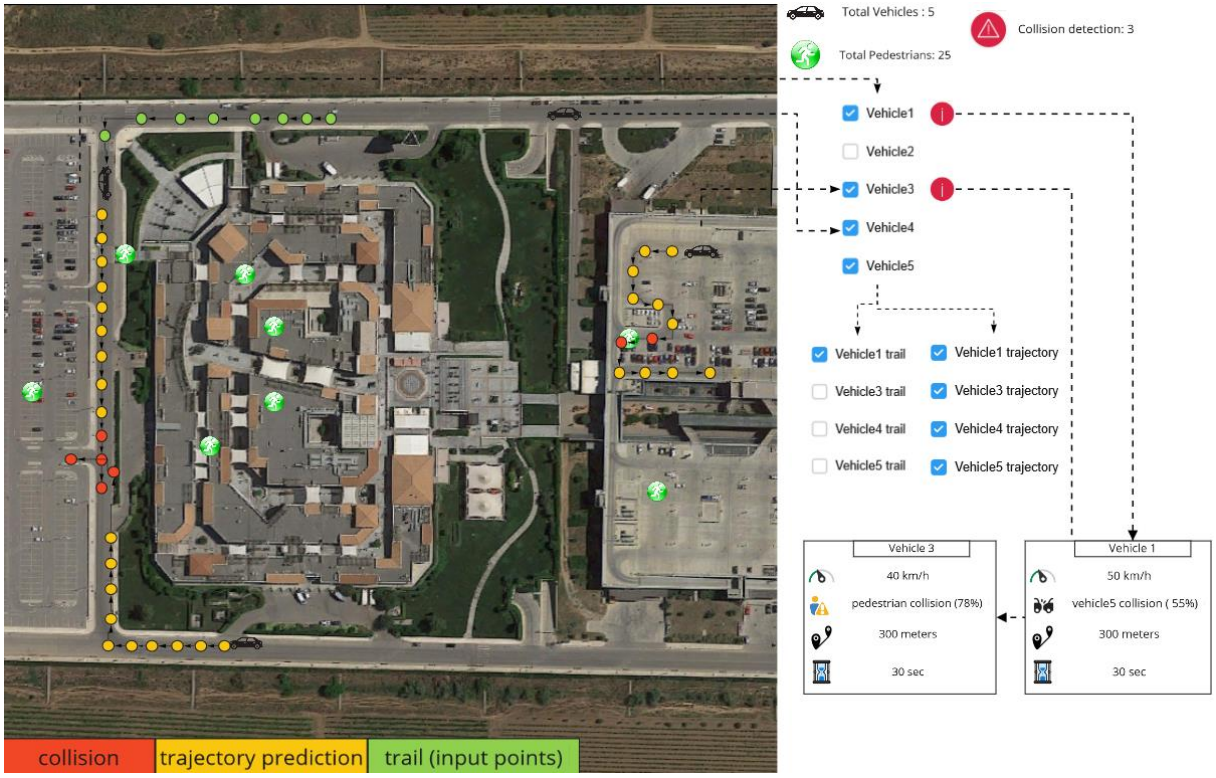

Figure 4.2 Vehicle trajectory and collision alerts

For the purposes of this PoC, the main service view is presented in Figure 4.2. In detail, the user can monitor the total number of vehicles, pedestrians and collision detections for a specific area. Additionally, the user is able to select which vehicle should be visible on map through a multiple selection list. After selecting a vehicle, two additional multiple selection lists appear for the actual trail points (used for the prediction) and predicted trajectory points. Furthermore, the user is given the option to select which of them will be displayed on the map in various combinations, e.g. trail of Vehicle1 and the trajectory points for all vehicles. For each of the selected vehicles, the following information are displayed on the right-hand side: average travel speed; collision information (in case that a collision is detected among vehicles and or pedestrians); distance remaining from collision point; and time remaining from collision.

4.1.3 *Demonstratable Scenarios*

In this demonstration scenario, the main purpose is to prove that the LOCUS Platform is able to facilitate the monitoring and detection of possible collisions, as well as to calculate their probability and estimate their severity.

The following steps will be part of the demonstration:

Step 1. On the main page, the user selects a specific area to see. The user selects some of the vehicles to be appeared on the map using a multiple selection list.

Step 2. For each vehicle, the user can see the trails (i.e. points in its route) and/or the trajectories of their routes which is part of the predicted route estimated using LOCUS functions for trajectory predictions and/or popular trajectories.

Step 3. Along with the trajectories, the user is also able to see, with different colour, the exact points on which the collision has been detected. Then, he/she clicks on the information button, next to each vehicle to see some further details, such as:

- Average speed of vehicle;
- Type of collision (e.g. collision with another vehicle, collision with a pedestrian;
- Probability of collision;
- Meters remaining from the collision point;
- Time remaining from the collision point.

4.2 PoC 1 - Network Planning and Redesign Using Hybrid Spatial Network Clustering

This PoC extension aims at utilizing location information in conjunction with actual network KPI measurements (provided by the LOCUS Data Collection function) in order to optimize the current antenna configuration of an existing venue towards the actual traffic density hotspots of underlying users. This implementation is based on [37] that was performed by Incelligent as part of our dissemination activities with respect to WP4, network planning and efficient redesign using location information. In order to further showcase the usefulness of this implementation, appropriate REST services will be deployed using the LOCUS SDK on top of the LOCUS Platform that will provide an "on-demand network analysis and redesign" graphical user interface that will indicate a set of actions that can optimize the network performance for example venues.

4.2.1 *Service Design*

This module of PoC 1 will showcase two secondary services (to control the input and output of the required metadata or the report information) and one primary service that will control the invocation of the algorithm as shown in Table 4.2.

Table 4.2 Network redesign module services

Service Name	Inputs	Process	Outputs
3GPP network configuration ingestion service	List of shapes and meta-data (site latitude/longitude, azimuth angle, vertical tilt) for the existing cells within the boundaries of a specific test zone.	This information will be moved into the internal storage locations (i.e. the LOCUS Persistence Module's appropriate table schemas) and will be available to the main service in order to be used as input for the identification of appropriate actions.	Confirmation of the successful ingestion of the shapes and meta-data information (syntax-checking and IDs of the insertion).
3GPP Network traffic density analysis and redesign service (Primary)	A geometrical boundary that will be used for a selection filter of underlying radio elements and user equipment device measurement data as input to the algorithm.	The service will use the direct connection with a) the meta data, b) the geoshape and c) the user equipment device historical data to feed the unsupervised learning algorithm. The first step is the identification of traffic hot zones (with new GeoJSON boundaries and profiling). The second phase is to compute the azimuth steering actions based on the hotspots and the existing azimuth configuration. The outcomes are then stored in a processed (JSON-based) structure.	The reference ID of the execution. It can be used to monitor the status and to acquire the final result from the dashboard service.

3GPP Network Reconfiguration dashboard result service	The reference ID of the service execution for which the user requests the final outcome (network redesign dashboard).	This service checks the persistence of the user-space microservice in order to find if the result is also completed (in JSON structure) and fetches it to the user or sends appropriate message.	JSON structure (outcome from the azimuth optimization) that will include new azimuth locations and detected traffic hot zones (or traffic holes)

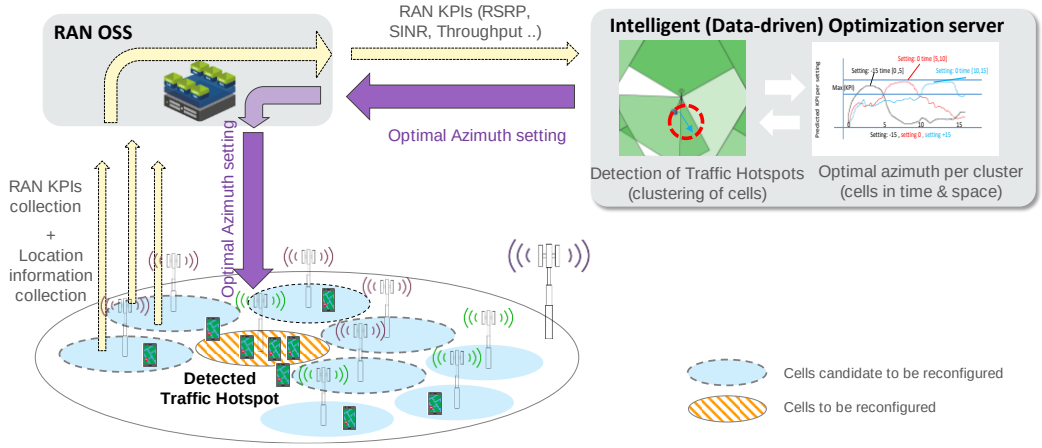

Figure 4.3 The process of optimization 3gpp network azimuth angles using location information and unsupervised learning

4.2.2 *Service View Analysis*

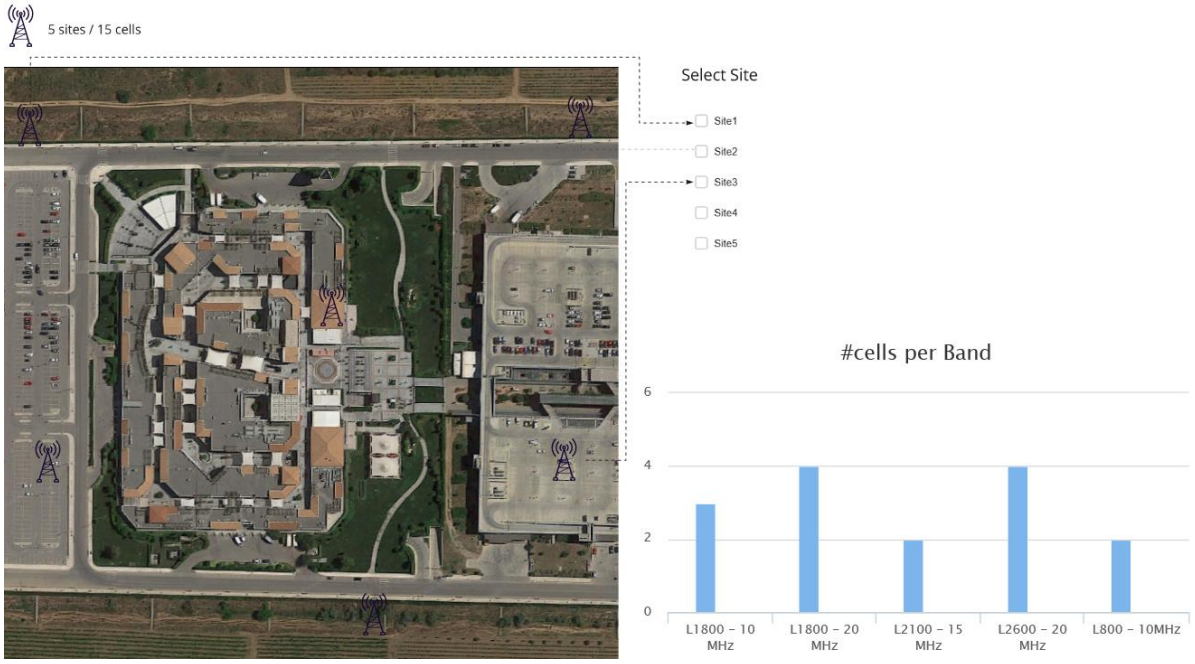

Figure 4.4 Area coverage default main page

Initially, in the service's main page (see Figure 4.4), the distinct number of sites and cells, which are installed on a specific area is presented. A multiple selection list of sites is available, from which the user can select one or more cells to be visible on the map. Moreover, a bar chart gives the user information about the frequency band of cells, which are available.

Then, by selecting a specific site, the user can focus on a specific cell -through a multiple selection menu- so that its coverage can be visible on the map. An information button is also available next to each cell, which -by clicking on it- offers details on the direction, azimuth, band and frequency of the cell in a pop-up window.

In addition to the above, a polygon appears on the map, specifying an area with low or no coverage, as shown in Figure 4.5. An alert message will then highlight the issue and will also suggest to the user to rotate the azimuth of some cells. This is done by clicking a button for performing the calculation.

After performing the aforementioned action, new shapes of different colour will appear on the map, showing the new coverage of the area. Once again, next to each cell the information about the azimuth rotation will be available (e.g., "Rotate this cell by 180° NW"). The latter are presented in Figure 4.6.

Figure 4.5 Low / no coverage detection on map

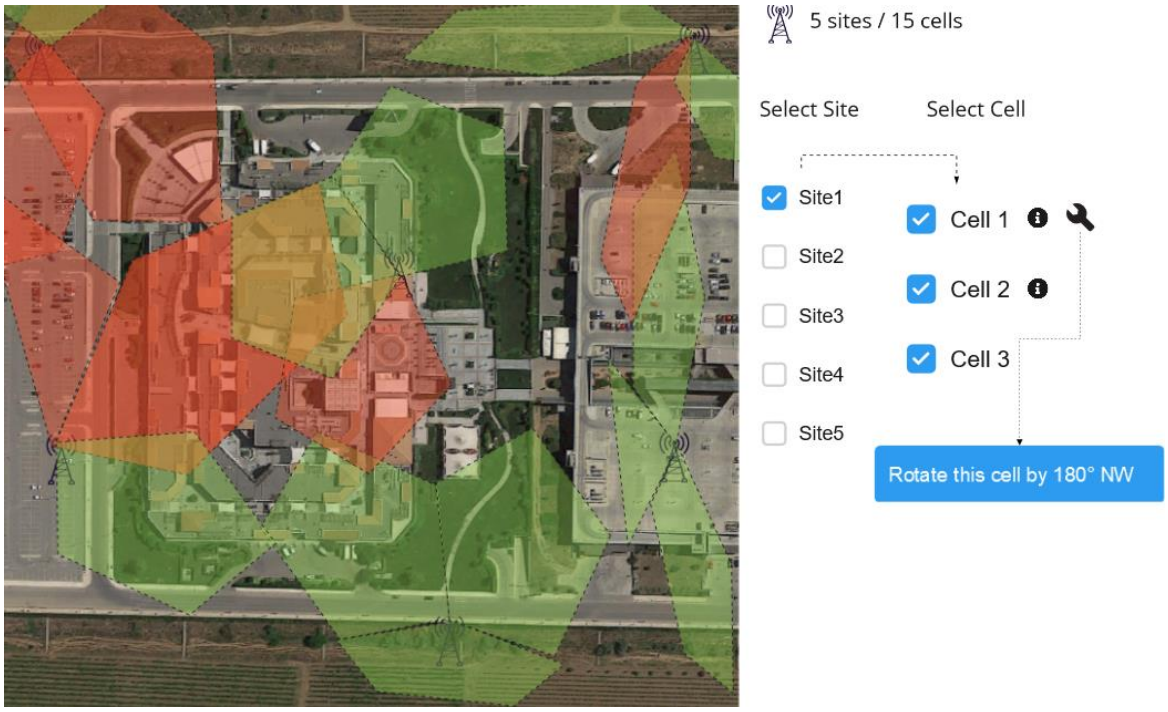

Figure 4.6 Azimuth rotation - new area coverage

4.2.3 *Demonstratable Scenarios*

The demonstration scenario refers to a case of performing actions on specific cells in order to improve coverage. These actions will be performed in an area confirmed as a hot spot with low and/or no coverage based on an ML/AI mechanism deployed on the LOCUS Platform and exposed to the network owner/3rd parties.

In more detail, the following steps will be performed in the demonstration.

Step 1. On the main page, the user selects a specific area to see. In the main page, the user can see some general information about the sites and cells:

- Position of cells on the map;
- Number of total sites and number of total cells on the selected area;
- Band and frequency of each cell;
- Azimuth and position of each cell.

Step 2. The user selects from a multiple selection list one or more sites. Then, selects the cells (one or more) for each site from a multiple selection list. The polygon of coverage per each cell is being appeared on the map. The user sees a warning message, which informs him or her that this polygon is a hot spot with low coverage accompanied with a suggestion to proceed with azimuth rotation by pressing a button. Hotspots are detected based on ML/AI mechanisms deployed on the LOCUS Platform, i.e. instantiating/ deploying specific LOCUS functions, at regular intervals and/or at the request of the telecom owner/3rd party through the service exposure mechanisms available in the platform.

Step 3. The user presses a button to perform a simulation of azimuth rotation on each cell that may affect the pre-defined polygon. As a result, a new map is displayed with the new coverage polygons of each cell. Next to each cell another option appears.

Step 4. The user can be informed about the azimuth rotation of each cell. The user decides which cell will rotate and how these actions will affect the entire area.

4.3 PoC 3 - Crowd mobility analytics

Crowd mobility analytics is presented as a use case in WP5 and specifically deliverables [31] [33] includes multiple analytics functionalities. The group inference service of the group mobility analytics is selected as a contribution to the PoCs due to its readiness level in terms of prototyping, implementation and testing. The service is based on using wireless traces to understand social group characteristics. The analytics system was previously implemented and tested against a ground-truth dataset from our controlled lab setup using Bluetooth beacons. Various possible application scenarios vary in the domain of smart cities and smart buildings. Example applications would include crowd management, tourism, and epidemic monitoring in a smart city. More detailed information about the analytics functionality, testing

results, and datasets can be found in [38] and the WP5 deliverables [31], [32]. Figure 4.7 illustrates the basic overview of the considered system. The people groups in the real world (left) can be detected using wireless scanners in the environment (middle) where the crowd mobility analytics classifies the people groups, and the analytics outcomes would be the number of people and number of groups in a given environment (right).

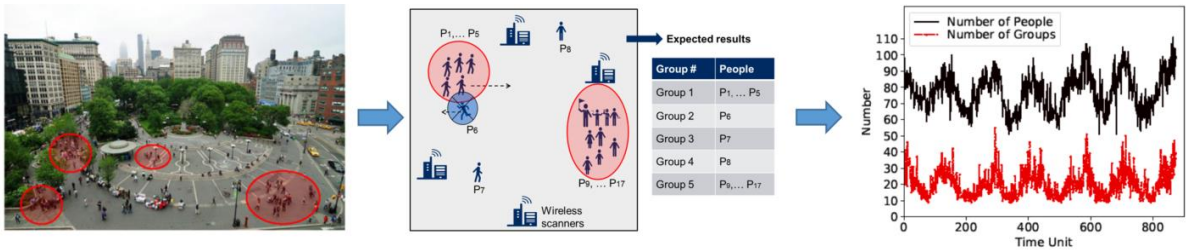

Figure 4.7 Crowd mobility analytics: Group inference in smart cities by the wireless traces of the mobile entities.

4.3.1 *Service Design*

The design of the service considers wireless measurements from multiple wireless scanners (e.g., Wi-Fi, Bluetooth, LTE base station) in the environment where mobile entities, such as people's mobile devices, are dynamically moving in the environment. The system is implemented through a multi-phased data pipeline (internal processes) and analytics and with different features such as centralized, decentralized computing and unsupervised machine learning through clustering techniques. These internal processes are shown in Figure 4.8.

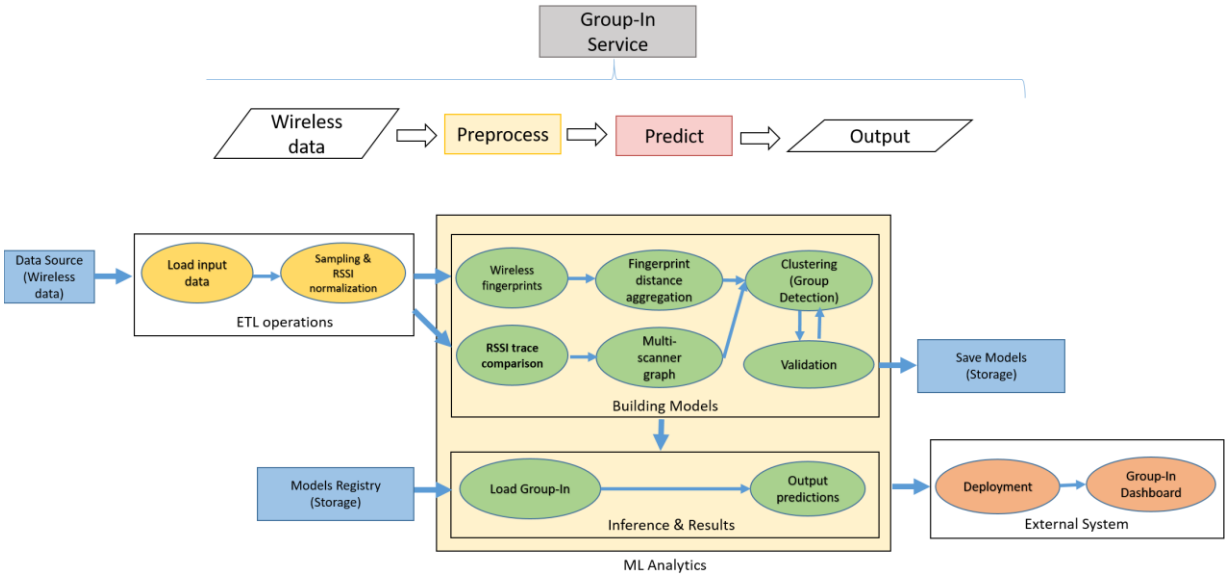

Figure 4.8 The ML-based service pipeline for the crowd mobility analytics

Table 4.3 describes the inputs, process, and the outputs for the crowd mobility analytics service for group inference.

Table 4.3 Crowd mobility analytics services

Service Name	Inputs	Process	Outputs
Group inference analytics function	Wireless (Bluetooth advertisement) packets as JSON documents. Input features: Timestamp, RSSI, Reference RSSI.	The analytics function consists of data pre-processing (sampling and RSSI normalization), graph model creation (centralized/ decentralized computing algorithms), and prediction through clustering.	File/REST API with JSON including the list of groups and group members inferred by the analytics function (including single person groups).
Group-In dashboard	Dashboard subscribes to the output files/ REST API from the analytics function.	The dashboard is a front-end process that accesses the output results from the server. It dynamically gathers analytics results.	Insights as visual graphs on the front-end.

4.3.2 *Service View Analysis*

The current service is implemented using Python; a web interface shows group mobility insights as shown in Figure 4.9. The dashboard can operate for historical data or online (live) data. For the historical data, the implementation allows for the emulation of a dynamic (real-time) dashboard by displacing iterations over the through historical time intervals.

The dashboard includes the number of groups (Figure 4.9, top) for the most recent time intervals. In detail, the dashboard lists the last nine time intervals, however, the dashboard can be configured to include more time intervals. The time interval itself is one of the parameters of the service. Based on the chosen time interval, the analytics functions will behave differently. The top figure presents a bar graph where the timeline is on the x-axis and the number of groups on the y-axis.

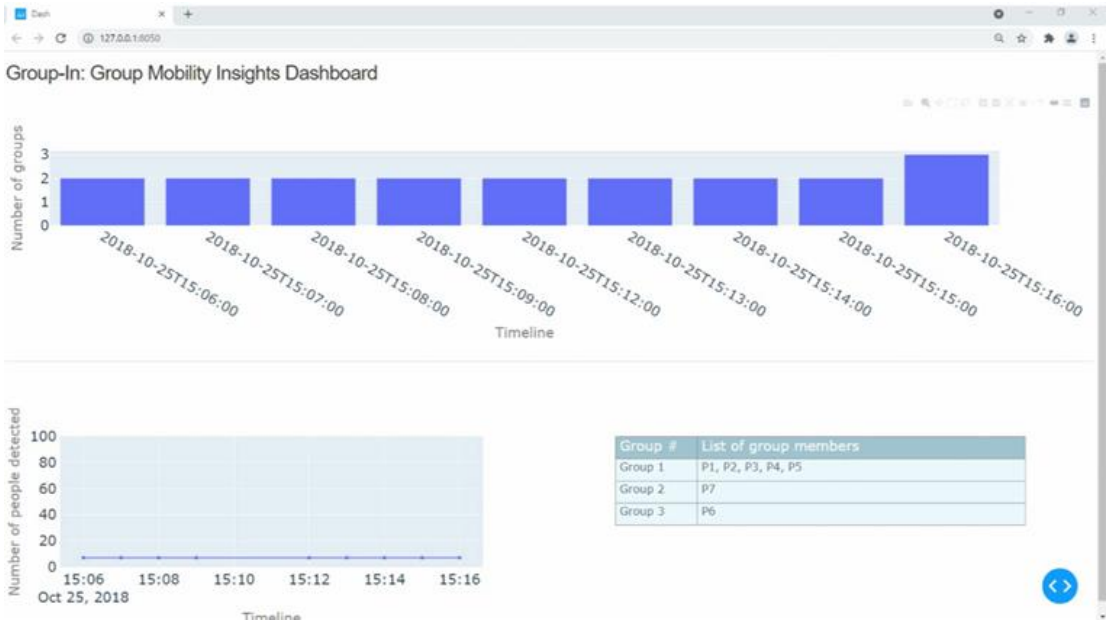

Figure 4.9 Group mobility insights dashboard.

In the bottom of the dashboard two additional results are listed. The bottom-left includes the total number of people in the given environment. The values are close to seven due to the limited scenario in the controlled setup, whereas in a real dataset the values reach more than 100 mobile entities inferred at a given time. On the right side, the list of the group members is listed in a table. For the simple example of the seven devices scenario, three groups are listed, where one includes five mobile entities and the other two include only one mobile entity. The list of group members refers to an anonymous set of members as no personal information is used in the system, where the anonymous IDs (i.e., 1, 2,.., 7) represent the mobile entities that are carried by individuals. In other words, the goal of the service and the visualization is to infer the number of groups and group sizes in a given environment as opposed to tracking an individual's movements.

4.3.3 *Demonstratable Scenarios*

As aforementioned, possible application scenarios include the application in smart cities and smart buildings. For the simplicity of deployment and ground-truth data collection, the dataset was collected in the lab environment where multiple office rooms were considered, and the movement behaviours of the entities were simulated by carrying a limited set of wireless beacon devices. The scenario was to test the analytics functions' accuracy and gather statistical insights from the system behaviour in terms of different clustering techniques, machine learning, the centralized and decentralized algorithms, and the effect of the system parameters such as sampling time and time interval for group inference.

Demonstrable scenarios will include the testing scenarios >20 tests conducted in the lab environment. These scenarios involve a static scenario where the devices are statically placed

in the groups in separate office rooms or in the same office rooms as well as mobile scenarios, where the devices dynamically change their locations in a controlled and uncontrolled manner. For instance, visualizing the system outputs and accuracy during the straight movements and random walking behaviour are included in the testing scenarios. For the static scenarios, it is expected to see that the groups do not change over time intervals. For the dynamic movement scenarios, as the devices dynamically change their locations, the number of groups and the group members may have fluctuations through the time intervals, especially for the scenarios where the analytics is shown not to perform well (e.g., when inter-group distances are ~1m). Similarly, from one scenario to another, the group members are expected to change due to each scenario having a different set of groups.

As possible work items in the upcoming months, more insights on the scenarios (e.g., visualizing which testing scenario and the expected outputs while listing the group inference analytics outputs) are also considered for visualizing further statistical insights from the analytics system and using different clustering and machine learning algorithms. Lastly, the testing may include integration to the system and the LOCUS dataset (University of Malaga testbed) which has a different set of data features and characteristics such as 4G LTE, Wi-Fi, and UWB features. The applicability of the new dataset to the existing system is currently under consideration.

5 Conclusion and Future Work

To sum up, this deliverable presents the current status of the LOCUS Consortium's demonstration activities. Partners have been steadily progressing in the work to be carried out until the final PoCs take place.

More specifically, following the work described in Deliverable 6.1 [2], partners have presented updates and more details on the use cases to be demonstrated, as well as introduced new UCs and scenarios that will help deliver a more complete image of the LOCUS Platform capabilities. For each use case, the services to be demonstrated are designed and detailed, while the updates in the services views, i.e. indicative UIs, have also been included in this document.

Furthermore, all the efforts towards the testbeds' setup at OTE and UMA premises have been described in detail, along with the System Architecture aspects and related infrastructure.

With respect to future work, partners will continue to finalize the LOCUS functions and related work to be demonstrated, i.e. scenarios, services/applications and UI components, including the adaptation of the aforementioned to the UMA testbed data model, since this testbed will provide the input data for some of the use cases. Additionally, the onboarding of all the LOCUS platform components and functions will be finalized in both testbeds, which will then be extensively tested at various stages before the final demonstration.

6 References

[1] Deliverable 2.5 EU LOCUS project “System Architecture: final version”

[2] Deliverable 6.1 EU LOCUS project “Detailed requirements from scenarios and application specification”

[3] Ubuntu, https://ubuntu.com/

[4] Queens Openstack, https://releases.openstack.org/queens/

[5] “Documentation for Xena (October 2021)”, https://docs.openstack.org/xena/

[6] Deliverable 4.3 EU LOCUS project “Implementation of the Virtualization platform for network control and management

[7] Deliverable 4.4 EU LOCUS project ““Implementation of the Virtualization platform for network control and management: final version”

[8] Openstack, https://www.openstack.org/

[9] Kubernetes, https://kubernetes.io/

[10] Rancher, https://rancher.com

[11] Open-Source MANO, https://osm.etsi.org/

[12] Docker, https://www.docker.com/

[13] Harbor, https://goharbor.io/

[14] RabbitMQ, https://www.rabbitmq.com/

[15] Bind9, https://www.isc.org/bind/

[16] Helm Chart, https://helm.sh/docs/topics/charts/

[17] Deliverable 3.3 EU LOCUS project “Integrated localization technologies: preliminary version”

[18] Deliverable 3.4 EU LOCUS project “Integrated localization technologies: final version”

[19] Deliverable D2.4 EU LOCUS project “System Architecture: preliminary version”

[20] Kompose, https://kompose.io/

[21] Apache Hadoop, https://hadoop.apache.org/

[22] Apache HiveTM, https://hive.apache.org/

[23] TrinoIO, https://trino.io/

[24] Consul, https://www.consul.io/

[25] Deliverable 3.1 EU LOCUS project “5G Based Localization Solutions, preliminary version”

[26] Deliverable 3.2 EU LOCUS project “5G Based Advanced Localization Solutions, intermediate version”

[27] Juju Charm, https://jaas.ai/

[28] ETSI GS NFV-SOL 006, “Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; NFV descriptors based on YANG Specification”, V2.7.1 (2019-12),

https://www.etsi.org/deliver/etsi_gs/NFV-SOL/001_099/006/02.07.01_60/gs_nfv-sol006v020701p.pdf

[29] A. Conti, F. Morselli, Z. Liu, S. Bartoletti, S. Mazuelas, W. C. Lindsey, and M. Z. Win, "Location awareness in beyond 5G networks," IEEE Commun. Mag., vol. 59, no. 11, pp. 22–27, Nov. 2021.

[30] Apache Kafka, http://kafka.apache.org/

[31] Deliverable 5.1 EU LOCUS project "Design and implementation of virtualization technologies and pattern recognition mechanisms for physical analytics, preliminary version"

[32] Deliverable 5.2 EU LOCUS project "Design and implementation of virtualization technologies and pattern recognition mechanisms for physical analytics - final version"

[33] Deliverable 5.3 EU LOCUS project "Design of the localization & analytics as a service solution"

[34] P. E. Hart, N. J. Nilsson and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," in IEEE Transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100-107, July 1968, doi: 10.1109/TSSC.1968.300136.

[35] Apache CouchDB, https://couchdb.apache.org/

[36] Y. Filippas, A. Margaris and K. Tsagkaris, "Deep Learning Approaches for Mobile Trajectory Prediction," 2021 IEEE Globecom Workshops (GC Wkshps), pp. 1-6, 7-11 Dec. 2021, doi: 10.1109/GCWkshps52748.2021.9682164.

[37] A. Margaris, I. Filippas, K. Tsagkaris, "Hybrid Network–Spatial Clustering for Optimizing 5G Mobile Networks", *Appl. Sci.*, vol. *12*, no. 3, p. 1203, 2022, doi: 10.3390/app12031203

[38] G. Solmaz, J. Fürst, S. Aytac, and F.-J. Wu, "Group-In: Group Inference from Wireless Traces of Mobile Devices.", In Proceedings of ACM/IEEE IPSN'20, April 2020.