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 D6.3

"Assessment of applications integrated with geolocation mechanisms"

Deliverable Type:	R
Dissemination Level:	Public
Contractual Date of Delivery to the EU:	31/10/2022
Actual Date of Delivery to the EU:	14/11/2022
WP contributing to the Deliverable:	WP6
Editor(s):	VIAVI, Takai Eddine Kennouche
Author(s):	VIAVI, Takai Eddine Kennouche, Karen Boulos
	UMA, Antonio Tarrías, Emil Jatib Khatib
	TEI Stefano Stracca, Marzio Puleri
	INCE, Athina Ropodi, Yannis Filippas, Aristotelis Margaris, Kostas Tsagkaris
	NEC, Gürkan Solmaz
	NXW, Nicola Venturi, Michael De Angelis, Giacomo Bernini

	CNIT, Stefania Bartoletti, Andrea Conti, Domenico Garlisi, Ivan Palamà, Gianluca Torsoli, Nicola Blefari Melazzi OTE, Maria Belesioti ORA, Sana Ben Jemaa, Imed Hadj Kacem IMDEA, Domenico Giustiniano, Giuseppe Santaromita
Internal Reviewer(s):	NXW, Giacomo Bernini OTE, Maria Belesioti SAMS, Oluwatayo Yetunde Kolawole CNIT, Stefania Bartoletti, Nicola Blefari Melazzi
Short Abstract:	The objective of this deliverable is the assessment of the technical achievements of the project with respect to the targets set in the previous deliverables. It also details the different results of the PoCs after their integration in the LOCUS platform. Furthermore, it presents the results of some external demonstrations that are not integrated in the LOCUS platform and gives a brief insight of the business value of the various achievements of the project.
Keyword List:	Use Case, PoC, storyline, geolocation, analytics

Executive Summary

The LOCUS project developed a set of use-cases that leverage geolocation information and extract analytics in support of network management and new services and verticals. Furthermore, the project implemented three Proofs of Concept (PoCs), integrating the LOCUS platform, with a selected set of use-cases from Work Packages (WP) 3, 4 and 5. The PoCs are implemented as pipelines of containerized applications, in a microservices architecture that is cloud-native. This deliverable focuses on the technical achievements and analysis of the PoCs, while detailing their integration in the LOCUS platform, with selected results. In addition, it summarizes additional demonstrations carried out in the project and separated from the Proof-of-Concepts.

The structure of this document is as follows. Section 2 is used to summarize the technical achievements of the project through an assessment of the platform, and of the different use cases as detailed in previous deliverables. Section 3 is dedicated to the PoCs. It is used to describe the PoCs results after their integration in the platform. It also recalls the description of the different storylines belonging to the PoCs and details some execution aspects not described in the previous deliverables. Section 4 summarizes the additional demonstrations carried out for a selection of localization enablers. The last section is used to conclude the document and gives a brief insight of the business value of the project.

VERSION CONTROL TABLE			
VERSION N.	**PURPOSE/CHANGES**	**AUTHOR(S)**	**DATE**
0.0	ToC	VIAVI	03/08/2022
0.8	Review-ready draft	See list above	04/11/2022
0.9	Edits and Corrections	See list above	08/11/2022
0.10	Consolidated, after internal reviews	VIAVI	11/11/2022
1.0	Overall Revision	CNIT	13/11/2022
1.1	Final Revision by the coordinator	CNIT	14/11/2022

Table Of Content

List of Abbreviations

5GC	5G Core	ONOS	Open Network Operating System
AGV	Automated Guided Vehicles	OPEX	Operating Expenses
AI	Artificial Intelligence	PMF	Probability Mass Function
AMQP	Advanced Message Queuing Protocol	POC	Proof Of Concept
AOD	Angle Of Departure	POI	Point Of Interest
CDF	Cumulative Distribution Function	PRS	Positioning Reference Signal
CPU	Central Processing Unit	RAN	Radio Access Network
DL	Down Link	RB	Resource Block
ECDF	Empirical Cumulative Distribution Function	RIC	RAN Intelligent Controller
GMM	Gaussian Mixture Model	RMSE	Root Mean Square Error
gNB	Next Generation Node B	RSRP	Reference Signal Received Power
GPS	Global Positioning System	RSRQ	Reference Signal Received Quality
GRU	Gated Recurrent Unit	RSSI	Received Signal Strength Indication
GUI	Graphical User Interface	RSTD	Reference Signal Time Difference
IDFT	Inverse Discrete Fourier Transform	SDR	Software-Defined Radio
IPG	Inter-Packet Gap	SON	Self-Organized Networks
KPI	Key Performance Indicator	SRS	Sounding Reference Signal
LBS	Location Based Service	TDD	Time Division Duplex
LEN	Localization Enabler	TDOA	Time Difference Of Arrival
LMF	Location Management Function	TOA	Time Of Arrival
LOS	Line-Of-Sight	UC	Use-Case
LSP	Location-based Service Provider	UDP	User Datagram Protocol
MANO	Management and Orchestration	UE	User Equipment
ML	Machine Learning	UI	User Interface

NFV	Network Function Virtualization	UL	Up Link
NLOS	Non-Line-Of-Sight	UWB	Ultra-Wide Band
NR	New Radio	VNF	Virtual Network Function
NSD	Network Service Descriptor	VR	Virtual Reality
OAI	OpenAir Interface	WP	Work Package
ONF	Open Networking Foundation		

Table Index

Figure Index

1 Introduction

The LOCUS project proposes a variety of use cases that rely on geolocation information. These use cases are categorized into four topics: 1) Location Security and Privacy, 2) Localization enablers, 3) Network Management, 4) New Services. In addition, the project proposes an architecture, for platform that provides the necessary capabilities for the integration of use-cases into end-to-end solutions.

Three Proof of Concepts (PoCs) were developed throughout the project's duration, each PoC demonstrates a set of storylines, and each storyline is a composition of services and functions implemented and integrated on the LOCUS Platform. Each storyline's pipeline involves a specific set of microservices that consume and produce data. For example, one use case of localization enablers must feed another use case for smart network management in a way to deliver a particular service such as coverage optimization, or network diagnosis for troubleshooting. It is worth noting that the use case of localization enablers must rely on another from location security and privacy category to introduce privacy for the PoC. Figure 1 is used to clarify the mapping among the use cases, the PoCs and the storylines.

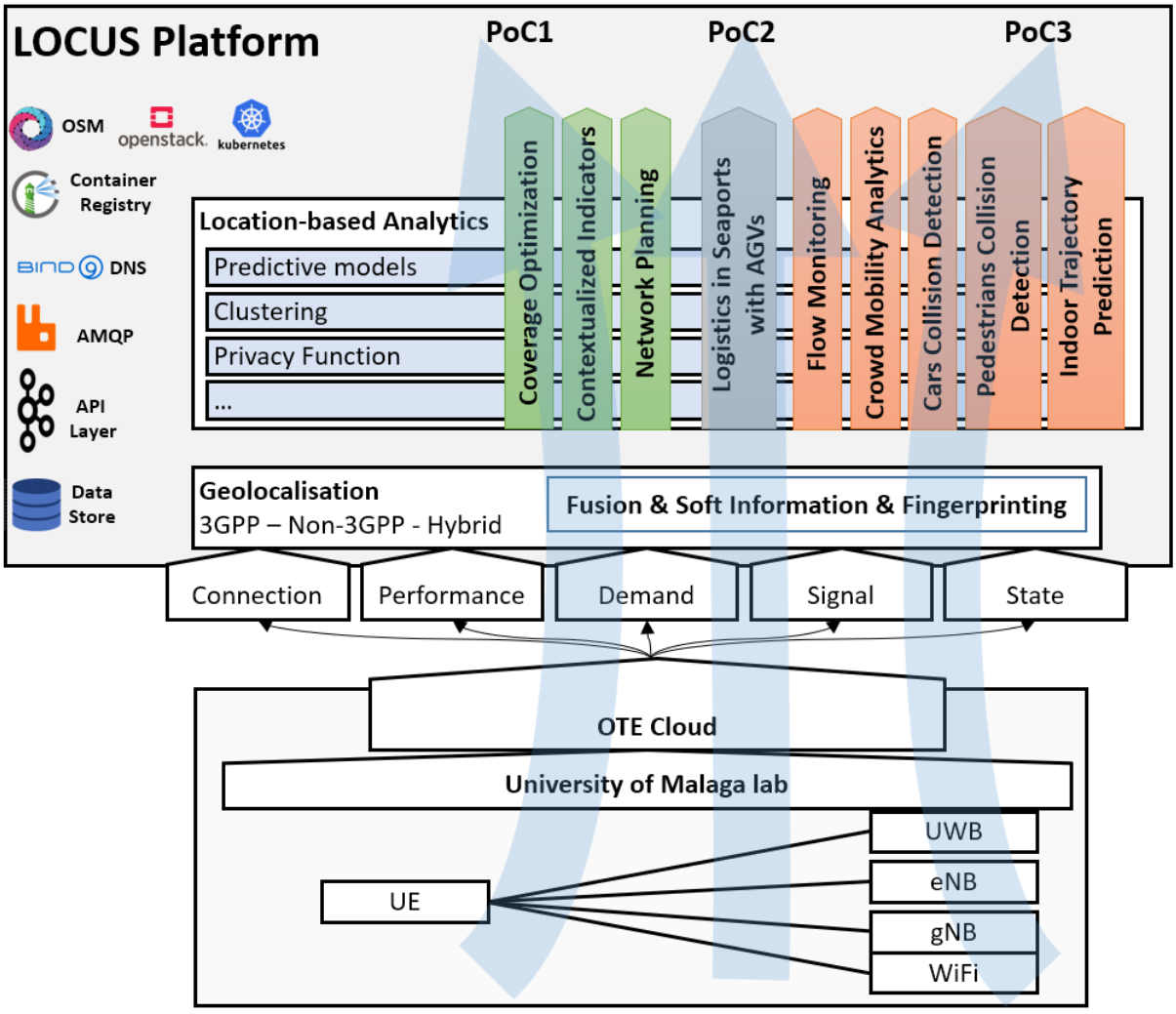


Figure 1: PoCs/ Storylines/ Use Cases

In the project, different use cases are proposed, building on localization information to build analytics and support new verticals. Use-cases are proposed to support smart network management, leading to reduction in Operational Expenditures (OPEX) of operators. Some other use cases are used to enable geo-powered verticals, such as flow monitoring inside large areas and mobility patterns recognition, which allow for an efficient management of a commercial area. The various project deliverables document the totality of the investigated use-cases, a few of which are selected to develop the PoCs. The selection is based on the use case utility, the level of maturity, feasibility based on partners interest and available resources. For the smart network management PoC, a use case for coverage optimization and another for network diagnosis has been selected from WP4, they introduce machine learning in coverage optimization on one hand, and a novel contextualized indicators approach that can support network troubleshooting and diagnosis on the other hand. PoC2 concerns a use case for logistics management at seaports using Automated Guided Vehicles (AGVs). This use case highlights the need of stringent delays for some applications which leverage the benefits of 5G networks to offer the required service. In PoC3, use cases that concern flow monitoring inside large areas have been also selected for integration. These use cases introduce efficient management for commercial areas and can increase revenues, optimize people and vehicles mobility, and extract value from location histories.

In Section 2, we provide additional Technical Achievements, highlighting some Platform-level results, location security and privacy, and Localization Enablers that support the PoCs. In Section 3, we dive into the PoCs, and document the integration effort and some of the obtained results. In Section 4 we provide information on additional demonstrations that were developed for specific capabilities and scenarios.

2 Highlights on Technical Achievements

2.1 Platform Level Achievements

This section reports on the final activities related to integration and validation of the LOCUS platform. In particular, it describes the implemented and tested workflows for enabling deployment and consumption of the localization analytics services supporting the various LOCUS use cases and PoCs. In addition, an assessment of the platform latency is performed, considering latency introduced by the LOCUS platform in the analytics service deployment and consumption operation.

These latest validation results have been achieved on top of the LOCUS platform deployed and integrated in the OTE testbed, which is depicted in terms of components (as described in [1] and [2]) in Figure 2.

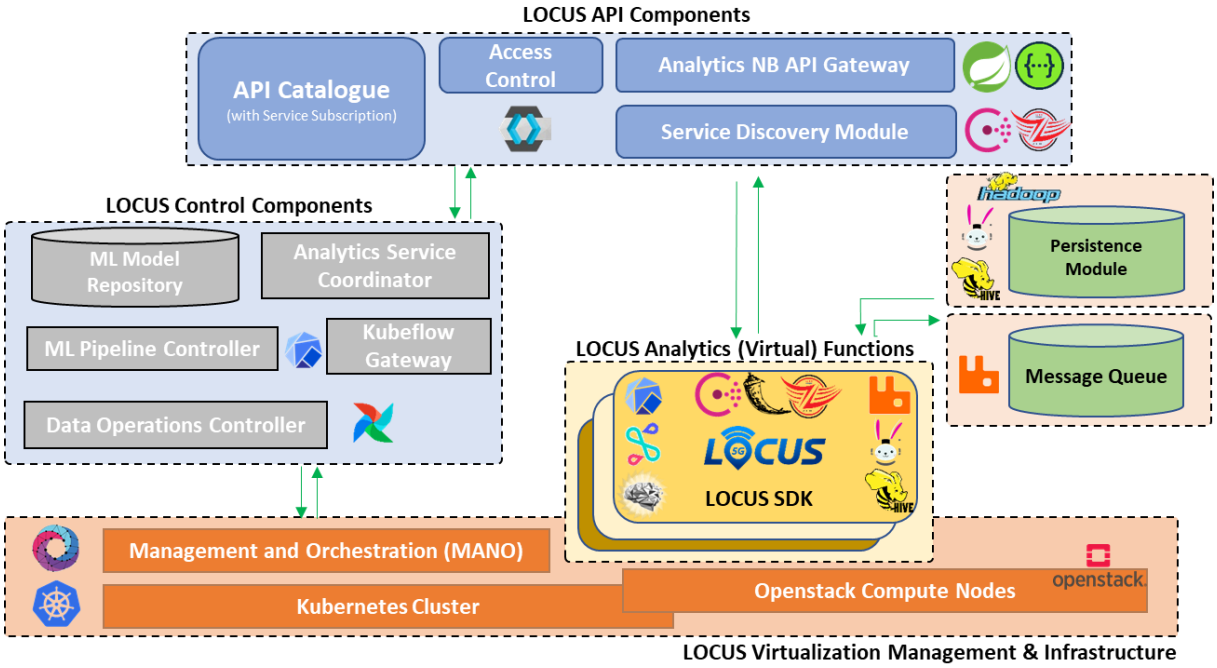


Figure 2: LOCUS platform components deployed in the OTE testbed

2.1.1 Technical integration of platform components

This section summarizes the main results achieved in WP6 concerning the integration and validation of the LOCUS platform components. Specifically, the following subsections reports the technical workflows enabled through the implemented LOCUS platform according to [1] (for the virtualization platform and LOCUS Management and Orchestration – MANO) and [2] (for the localization analytics as a service part, mostly built by the integration of the LOCUS API layer with the LOCUS platform control).

2.1.1.1 Analytics service deployment

The latest version of the LOCUS platform deployed in the OTE testbed has fully integrated the main LOCUS API layer and platform control components involved in the automated deployment of analytics services. First, the API Catalogue (embedding the service subscription capabilities), the Analytics Coordinator, the API Gateway and the Service Discovery module (implemented through Consul [3]). The integration for these components follows the specification highlighted in [4] and [2].

The integration illustrated in Figure 3, shows that it is possible to execute a complete workflow for the activation and the consumption of analytics services.

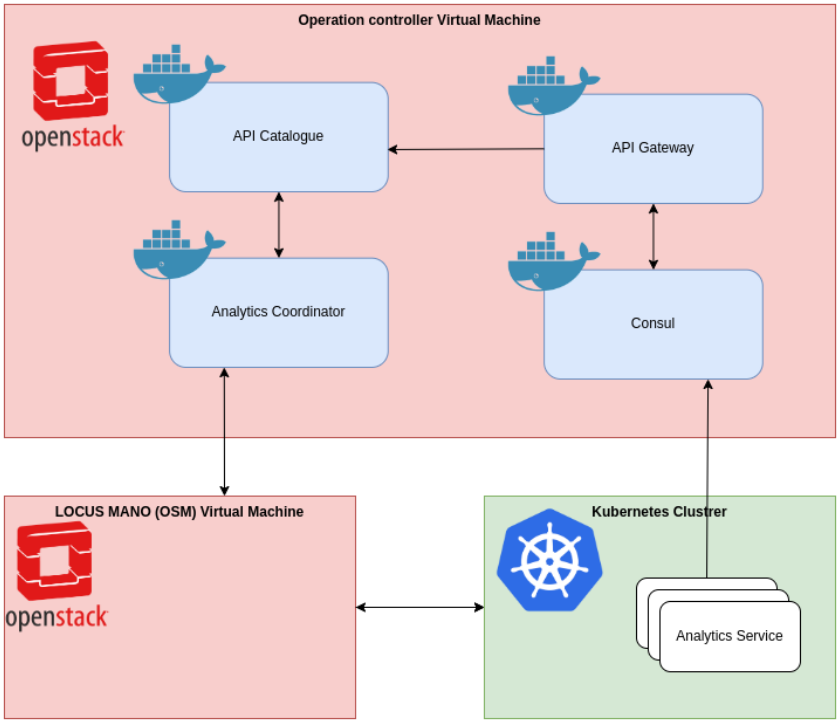


Figure 3: Components integration in LOCUS Platform

Figure 4 shows the workflow for the activation of a new analytics service after that has been onboarded within the API Catalogue.

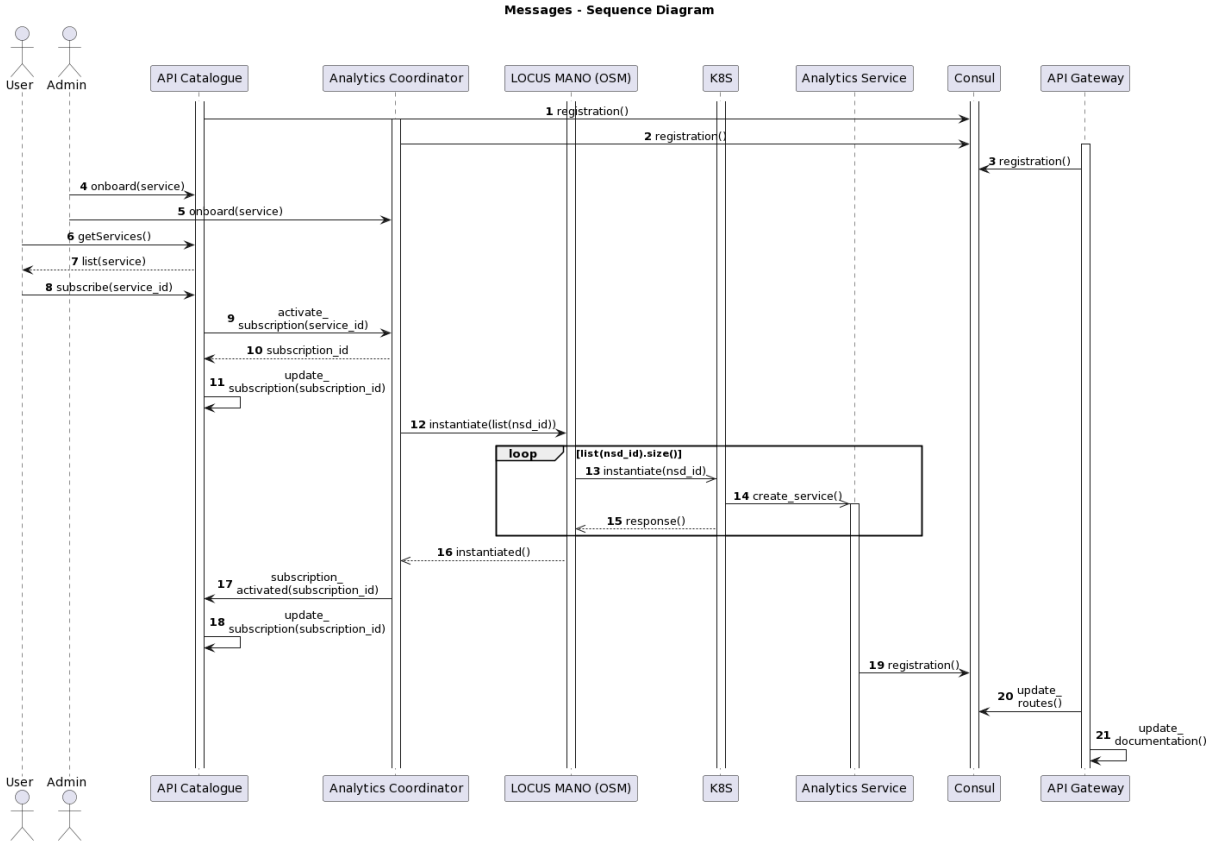


Figure 4: Analytics Service Instance subscription workflow

Below, a detailed description of Figure 4:

- Actions from 1 to 3: running Platform components register to Consul, the discovery service integrated within the LOCUS Platform.
- Actions 4 and 5: an administrator user onboards within the API Catalogue and the Analytics Coordinator Catalogue the required templates that describe the analytics service. These templates contain information about the service with a different level of abstraction that corresponds to the requirements needed by the API Catalogue and Analytics Coordinator to activate the service following the data model described in D5.3 [4].
- Actions 6 and 7: the user retrieves all the services available in the LOCUS Platform in order to choose the one to be activated.
- Action 8: the Platform user, to activate the service, sends a subscription request that contains the analytics service identifier to the API Catalogue and the activation information as per data model specified in D5.3 [4];
- Actions 9, 10 and 11: the API Catalogue validates the requests and forwards it to the Analytics Coordinator that sends back as response a UUID that corresponds to the service subscription identifier, which the API Catalogue uses to update the analytics service subscription record within its database.
- Action 12: the Analytics Coordinator, to allow the complete instantiation of the analytics service, decompose the analytics service requests into the constituent NFV Network Services that realize the actual service (in terms of Network Service Descriptors - NSDs). Then it sends an instantiation request for each Network Service that compose the service to the LOCUS MANO.
- Actions 13, 14 and 15: LOCUS MANO takes care to deploy the required Kubernetes Network Functions (KNFs, as per D4.4 [1]) into the Kubernetes (K8S) according to the service requirements expressed into the selected NSDs, and configures with Day1 procedures (as described on[1]) the created KNFs.
- Actions 16, 17 and 18: Analytics Coordinator, through a polling mechanism, receives the notification about the instantiation of the service by the LOCUS MANO and, after updating its internal database, forwards the notification to the API Catalogue that in turn updates its analytics service subscription record.
- Action 19 (goes in parallel with actions above): after the service and KNFs configuration, the deployed virtualized analytics service registers to Consul (according to the LOCUS SDK capabilities) to expose its analytics consumption APIs.
- Actions 20 and 21: the API Gateway interacts with Consul to update the internal routes that allow to forward the request to the newly deployed analytics service and exposes the corresponding API documentation to enable visibility and access of the analytics data consumption API offered by the analytics service.

2.1.1.2 Analytics service consumption

This section describes how to consume the analytics services instantiated by a LOCUS Platform user and available on the LOCUS Platform through the API Gateway. To allow the consumption of an analytics service, the API Gateway (based on Swagger [5]) exposes only the documentation of the services that are registered to Consul (Service Discovery Module) [3] checking with fixed periodicity if

new services are registered or old ones are unregistered (Actions marked with step “0” in the figure below, prior to consumption).

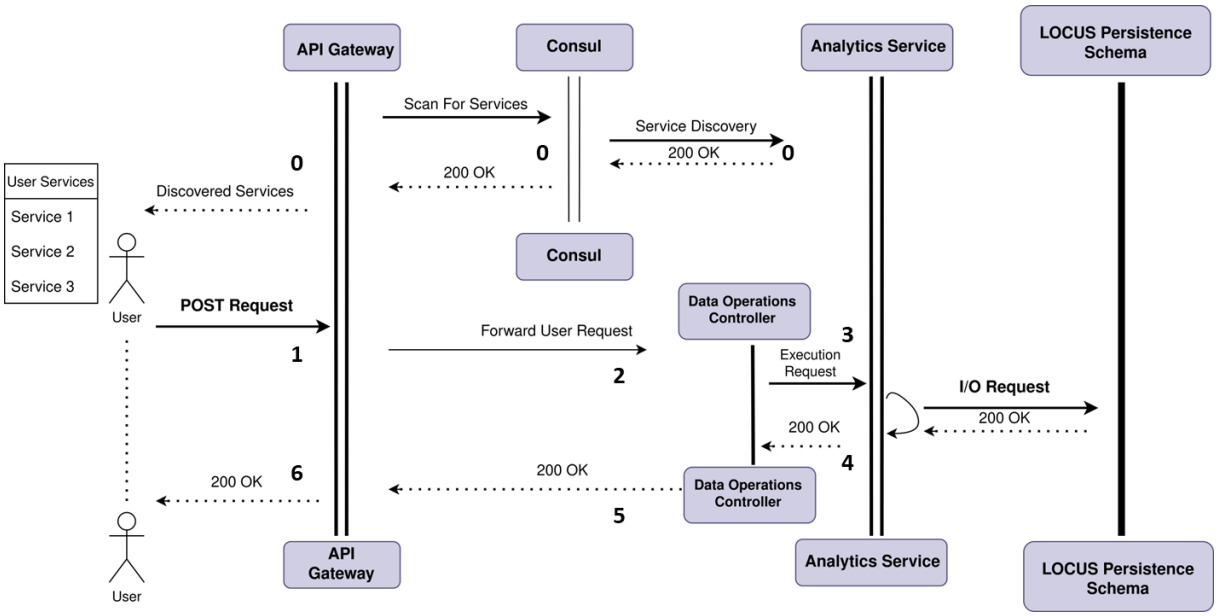


Figure 5: Analytics Service consumption

Figure 5 depicts the workflow for consuming an analytics service. A detailed description follows:

- Action 1: A Platform user interacts with the Swagger documentation of the API Gateway and triggers the desired endpoint of the chosen analytics service.
- Actions 2 and 3: the API Gateway thanks to Consul forwards the incoming request to the right analytics service that elaborates the request.
- Actions 4, 5 and 6: the selected analytics service produces its response (in the figure by querying data stored in the LOCUS persistence module) that is forwarded back to the API Gateway in order to be available to the Platform user.

2.1.1.3 Machine Learning pipeline on-demand execution

With respect to the LOCUS platform control components described in D5.3 [4] and D5.4 [2], a new component has been developed and integrated to enable on-demand execution of Machine Learning pipelines, and for example allow platform users (including administrators) to trigger training executions. Specifically, the Kubeflow-Gateway, shown in Figure 6, is a python-based FastAPI RESTful web application and it is a component developed in the context of the LOCUS platform in order to create a seamless link between the LOCUS Machine Learning Pipeline Controller, i.e., Kubeflow, and the LOCUS API Gateway. The main concept is to provide to the end user a unified interface to interact with the LOCUS Machine Learning Pipeline Controller and execute pipelines to train Machine Learning models in a standardized way through the LOCUS API Gateway. In order to do that, Kubeflow-Gateway registers to the LOCUS Service Discovery (Consul) so it can be advertised to the API Gateway that expose the northbound interface of the Kubeflow-Gateway itself through its living API specification. Pipelines executed in this way produce Machine Learning models that are stored in a MinIO object storage instance made available by Kubeflow (implementing the Machine Learning model repository). Figure 6 depicts the Kubeflow-Gateway and its main relations with the LOCUS platform components in the OTE testbed.

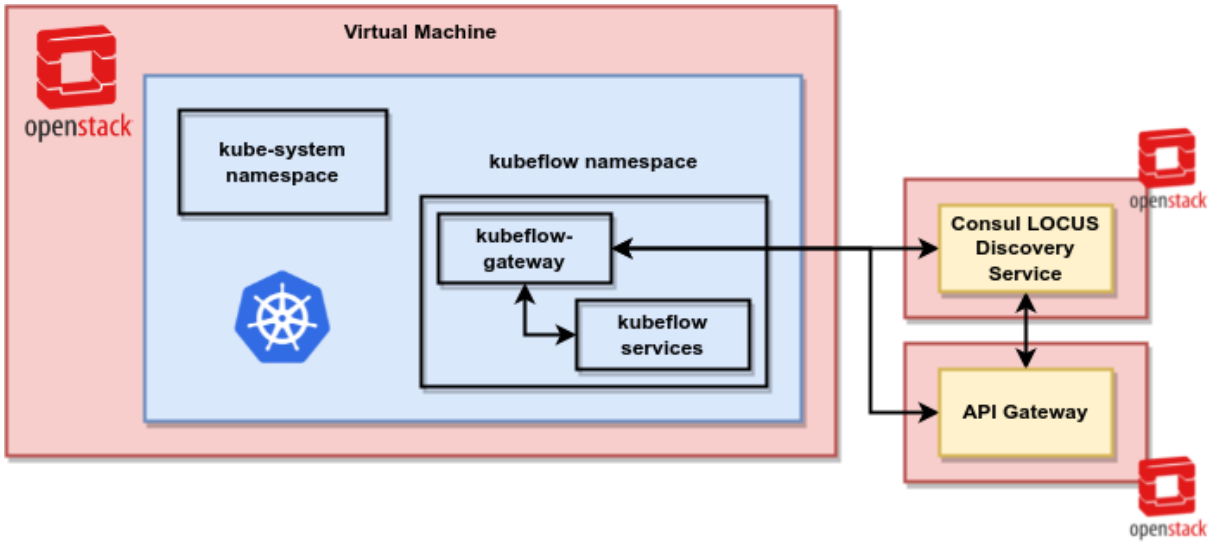


Figure 6: Kubeflow-Gateway within LOCUS Platform

The Northbound interface of the Kubeflow-Gateway is composed of the following endpoints.

Table 1: Kubeflow-Gateway APIs

Resource	HTTP Method	Description
/health	GET	Endpoint that returns the liveness of the Kubeflow-Gateway to the LOCUS Consul discovery service.
/train-model	POST	Start the execution of a pipeline available in the LOCUS Machine Learning Models Engine, specifying the identifier of the pipeline to be executed, the dataset url to be used to train the machine learning model and the filename of the resulting machine learning model itself.
/get-pipelines	GET	List the pipelines available in the LOCUS Machine Learning Engine, i.e., Kubeflow.
/get-models	GET	List the models trained through the pipelines on-boarded in the LOCUS Machine Learning Engine and stored in the MinIO instance.

Figure 7 depicts the swagger interface of the Kubeflow-Gateway accessed through the living swagger interface of the API Gateway in the OTE testbed.

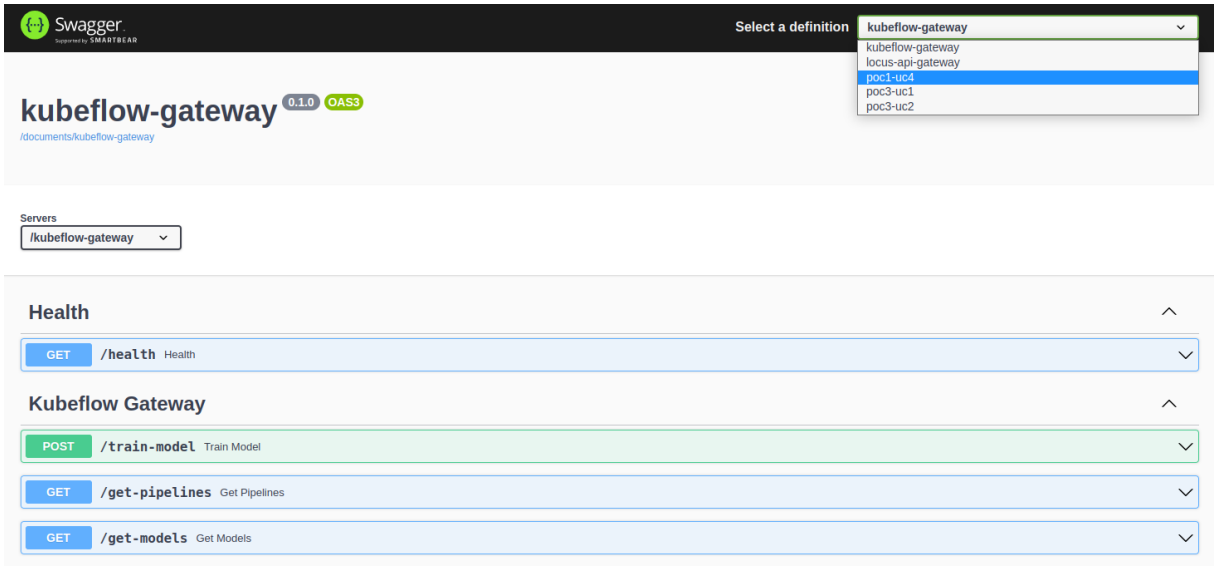


Figure 7: Kubeflow-Gateway swagger UI from API-Gateway

Figure 8 depicts the sequence of the actions performed for the execution of a pipeline on-boarded on the LOCUS Machine Learning Pipeline Controller.

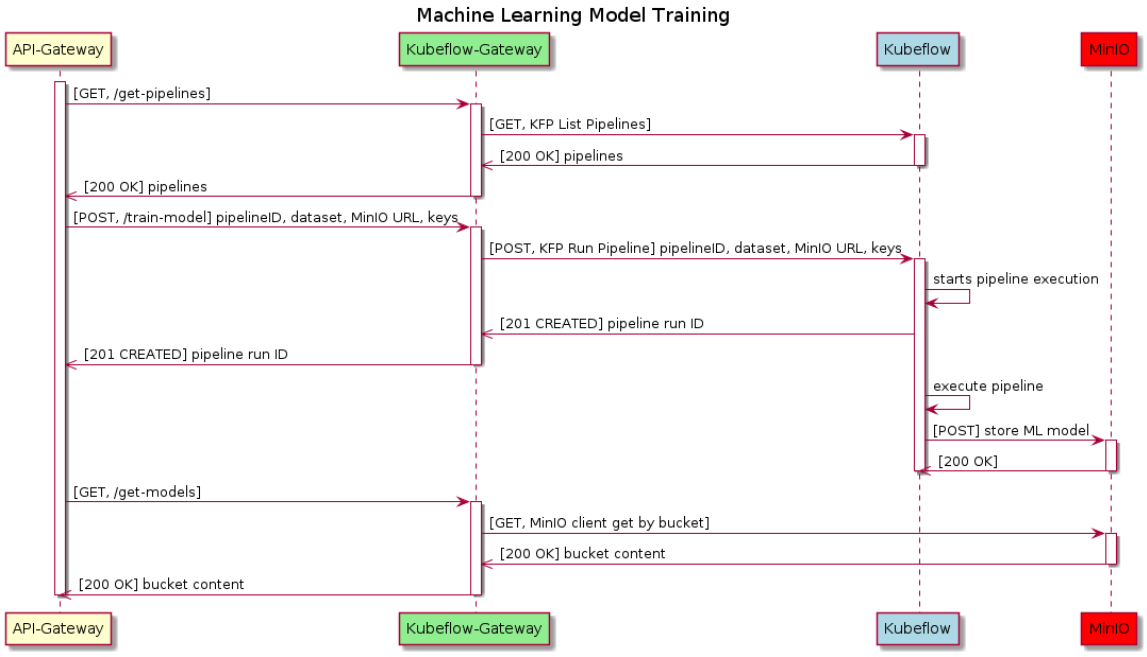


Figure 8: Kubeflow-Gateway model training workflow

Kubeflow-Gateway has been tested and validated with two different pipelines, the Soft Information-base Localization for a generative Model Training (D5.3 [5]) and the Trajectory Clustering Pipeline. The tests consisted in the execution of the pipelines through Kubeflow-Gateway in order to train and store the Machine Learning models described by the pipelines themselves. Figure 9 depicts the payload of the request performed to the Kubeflow-Gateway for the execution of the Soft Information-base Localization for a generative Model Training pipeline.

```
{
  "pipeline_id": "8bd338b9-6f8c-40fe-9b60-6d1cec5934bf",
  "dataset_url": "https://gitlab.com/GianlucaTorsoli/dataset/-/raw/main/input_gen_model_lte.csv?inline=false",
  "model_filename": "SI-model"
}
```


Figure 9: Kubeflow-Gateway payload for the execution of the Soft Information-base Localization for a generative Model Training

Figure 10, instead, shows the pipeline fully executed.

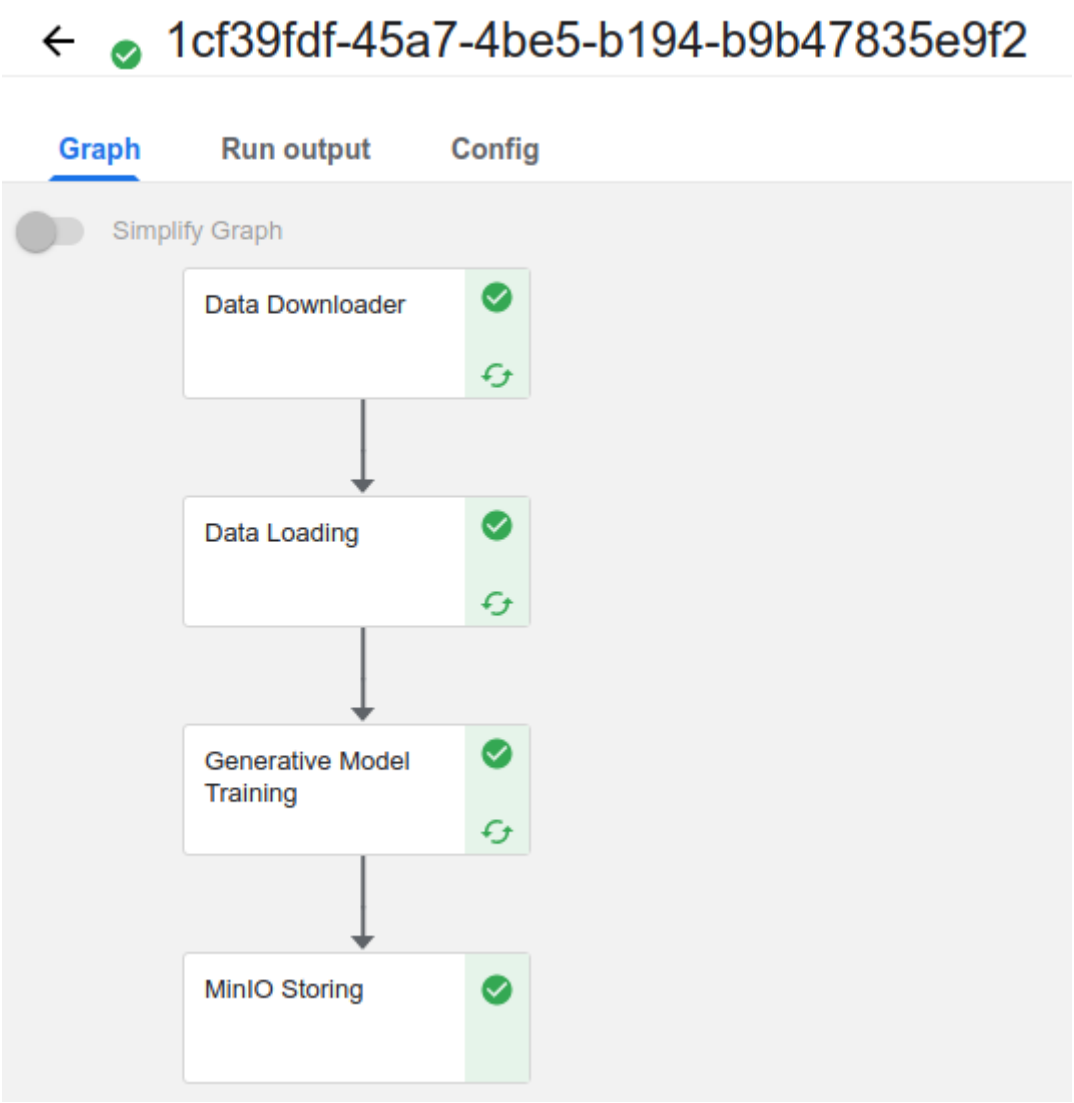


Figure 10: Execution of the Soft Information-base Localization for a generative Model Training pipeline

Finally, Figure 11 depicts the models available in the MinIO instance associated with the Kubeflow-Gateway; we can appreciate the presence of the model trained with the execution of the pipeline reported above.

```
[
    {
        "_bucket_name": "models",
        "_object_name": "lte",
        "_last_modified": "2022-10-07T12:39:42.144000+00:00",
        "_etag": "7b2fa812f6d1975e85a9b35b35d1636a-1",
        "_size": 2092,
        "_metadata": {},
        "_version_id": null,
        "_is_latest": null,
        "_storage_class": "STANDARD",
        "_owner_id": "",
        "_owner_name": "",
        "_content_type": null,
        "_is_delete_marker": false
    },
    {
        "_bucket_name": "models",
        "_object_name": "lte-test",
        "_last_modified": "2022-10-07T12:50:25.484000+00:00",
        "_etag": "560beff88e7c8223bbdfea35af4beae3-1",
        "_size": 2092,
        "_metadata": {},
        "_version_id": null,
        "_is_latest": null,
        "_storage_class": "STANDARD",
        "_owner_id": "",
        "_owner_name": "",
        "_content_type": null,
        "_is_delete_marker": false
    },
    {
        "_bucket_name": "models",
        "_object_name": "test-ldaf",
        "_last_modified": "2022-10-06T15:21:48.200000+00:00",
        "_etag": "e66764efe9623c6e57351bc333c308d3-3",
        "_size": 10781880,
        "_metadata": {},
        "_version_id": null,
        "_is_latest": null,
        "_storage_class": "STANDARD",
        "_owner_id": "",
        "_owner_name": "",
        "_content_type": null,
        "_is_delete_marker": false
    }
]
```


Figure 11: Machine Learning models available in MinIO instance

2.1.2 Platform Latency Assessment used as a metric for evaluation

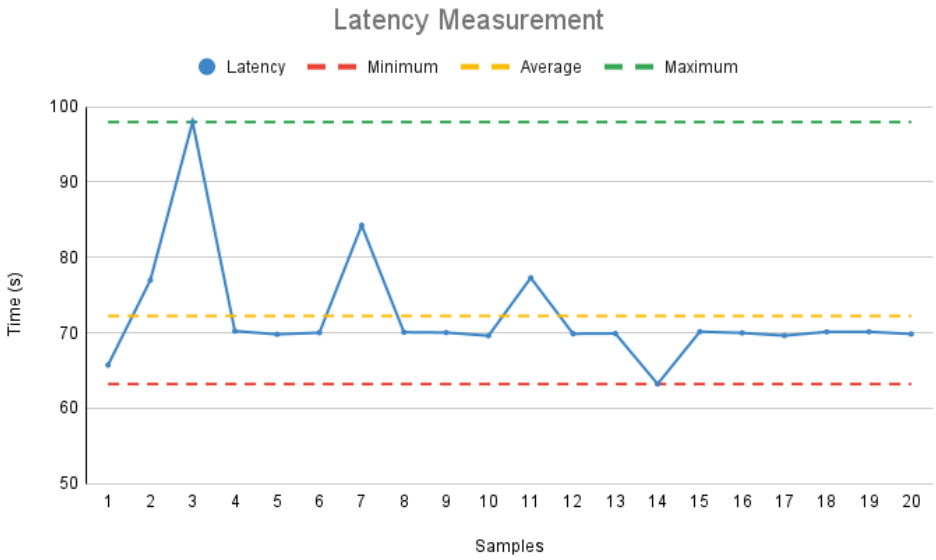
This section presents the latency assessment performed to certify the LOCUS Platform capabilities. Two different types of latency assessments are executed: latency assessment for analytics service instantiation and latency assessment for analytics service consumption.

2.1.2.1 Analytics Service Instantiation

The latency assessment of analytics service's instantiation has been performed by measuring the time between the sending of an instantiation request for a specific analytics service and the check that the service itself is instantiated.
For this type of assessment, three different types of trials have been conducted. Each trial is executed to generate 20 samples, all trials involve the instantiation of the analytics service using a single NSD.

The first trial consists of 20 sequential instantiations of the analytics service. The script that has been used launches an analytics service instantiation request from the API Catalogue (and specifically in the service subscription part) and then waits polling until the analytics service results in a "READY" status in the API Catalogue. The chart shown in Figure 12 describes the result of the first trial with a maximum time of instantiation equals to *97,97599077* seconds and a minimum equals *63,1993463* seconds.

Figure 12: Latency assessment for twenty sequential activations



The second measurement consists of twenty parallel instantiations of analytics service. The second script launches parallel requests equal to the CPUs that belong to the device used to run the script and then waits for the analytics services to be ready in a parallel manner. In our case the number of CPUs is four, so the script launched four parallel requests five times to cover the twenty requests needed. The following Figure 13 depicts the results.

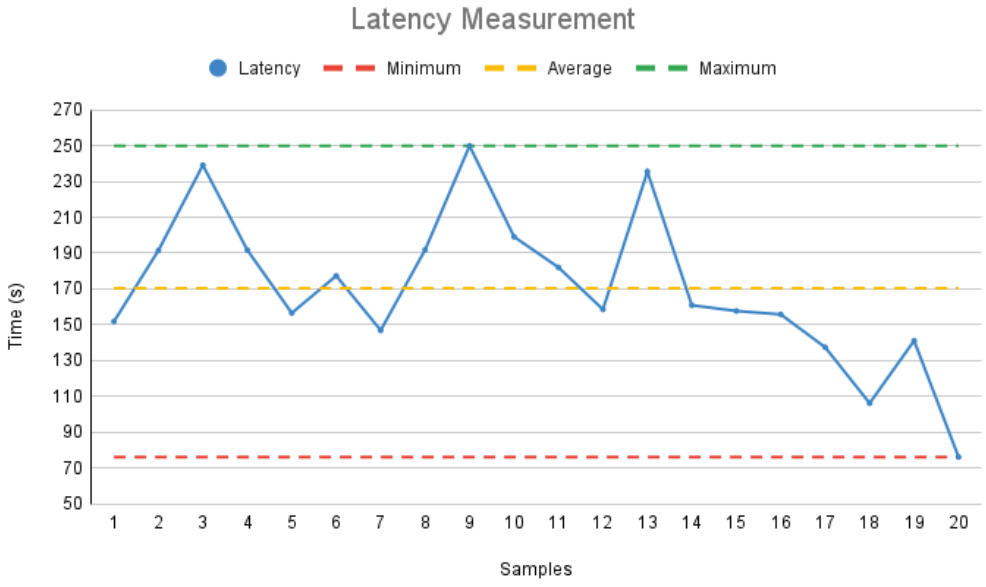


Figure 13: Latency assessment for twenty parallel activations with 4 CPUs waiting readiness

Like the second one, the third measurement consists of twenty parallel instantiations of analytics service, but with a different way to run the requests. The third script sends a number of parallel requests equal to the number of CPUs of the device used to run the script, but in this case, it does not wait for the readiness of the service and goes on with the other remaining requests until all the twenty are sent. Then it waits for the readiness of the analytics service. Figure 14 shows the result of the last trial.

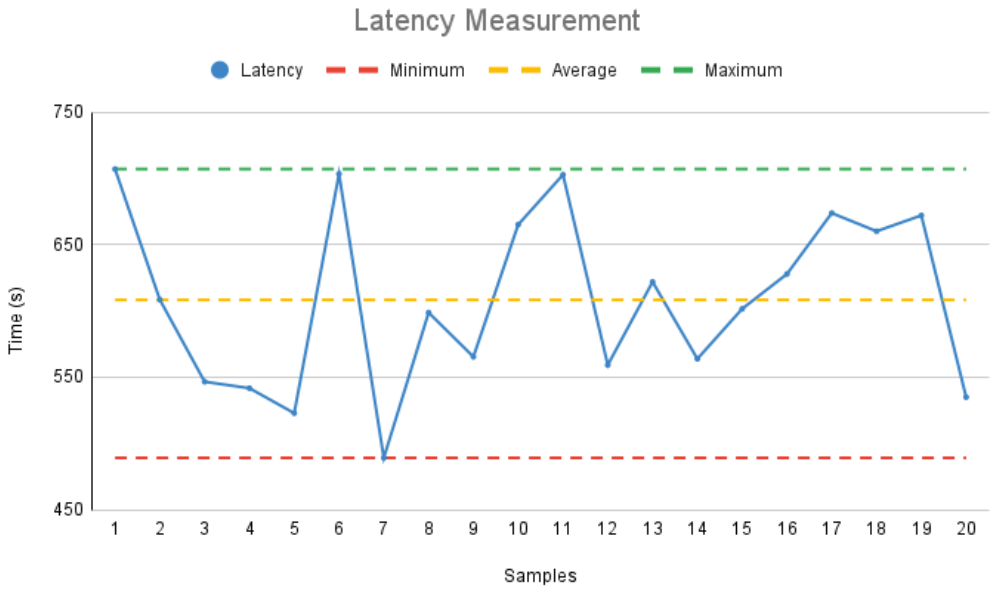


Figure 14: Latency assessment for twenty parallel activations with 4 CPUs without waiting readiness

It can be observed from the three charts above that although the activation time in the trials increase with the growing workload, the LOCUS platform supports the activation of the analytics service in an efficient manner.

2.1.2.2 Analytics Service Consumption

In order to understand the scale of the latency involved, a set of indicative measurements were performed that relate to the relevant platform component processes presented in Figure 5, as well as some example functions with variable sizing aspects in terms of dataset size.

Although it is not part of the actual service consumption, Action 0 was also measured.

The actual platform component overhead for the aforementioned API and Control blocks components are presented in Table 2.

Table 2: API and Control blocks latency measurements for service consumption-related processes[1]

Action	Description of measurement	Latency (ms)
0	API Swagger asks Consul and Consul lists registered services	25.8
1	A platform user that interacts with the Swagger documentation of the API Gateway, triggers the desired endpoint of the chosen analytics service	**10 & 13** (python & Java services respectively)
2-3	The API Gateway thanks to Data Operations Controller triggers the corresponding analytics service pipeline that serves the request	**20**
4-6	The selected analytics service forwards its response back to the API Gateway in order to be available to the platform user	**12 & 19** (python & Java services respectively)

In addition to the above, assuming an asynchronous training/ inference phase for the various ML approaches, the actual analytics service latency adding to the consumption phase is centred around the process of writing/persisting information and reading the results to be exposed. Indeed, such selected measurements are presented in the following table:

Table 3: Selected latency measurements for read/write operations in the LOCUS Persistence module for various tables, table sizes, and functions[2].

Operation/ process performed	Table Name in LOCUS Persistence Module[3]	Related Service/ Function[4]	No of rows	No of columns	Latency (s)
Read from LOCUS Persistence Schema	locus_geoshape	Correlation	1000	4	22.30
Read from LOCUS Persistence Schema	locus_location_viavi	Correlation	461	5	0.30
Store to LOCUS Persistence Schema	locus_correlation	Correlation	7059	6	2.58

[1] Measurements were performed using 2 containers with the following specifications: 512MB RAM & CPU 2 cores for the Consul container; and 8GB RAM and 2 cores for the API Gateway container.

[2] The hardware specifications for the virtual machine used for those measurements are 32 GB RAM and CPU Intel Xeon Processor (Skylake, IBRS) 16-core, 2.1GHz.

[3] Suffixes "_uma" and "_viavi" refer to the datasets employed and ingested according to the LOCUS schemas in the Persistence Module for testing/PoC purposes and specifically to the UMA testbed-related measurements and the OTE/VIAVI software -related data respectively.

[4] Detailed descriptions of the actual functions/services employed for measurements can be found in the PoC sections and specifically Section 2.3.

Read from LOCUS Persistence Schema	locus_location_viavi	Path Detection	1000	5	0.31
Store to LOCUS Persistence Schema	locus_geoshape	Path Detection	77	6	14.74
Store to LOCUS Persistence Schema	locus_collision	Collision Detection	7059	6	1.41
Read from LOCUS Persistence Schema	locus_geoshape	Path Detection	6982	1	0.83
Store to LOCUS Persistence Schema	locus_velocity	Path Detection	9309	4	663.21
Store to LOCUS Persistence Schema	locus_trajectory	Fingerprinting	683	19	114.44
Store to LOCUS Persistence Schema	locus_collision	Collision Detection	3610	7	94.92
Read from LOCUS Persistence Schema	locus_geoshape	POI Detection	6598	1	1.16
Store to LOCUS Persistence Schema	locus_trajectory	Trajectory Prediction	325	5	109.44
Read from LOCUS Persistence Schema	locus_location_uma	Fingerprinting	683	18	1.44
Read from LOCUS Persistence Scheme	locus_location_uma	Fingerprinting	683	18	0.20
Read from LOCUS Persistence Schema	locus_location_viavi	POI Detection	1032253	5	16.87
Read from LOCUS Persistence Schema	trajectories_out	Trajectory Prediction	6175	5	0.66
Read from LOCUS Persistence Schema	uma_data_processed	Trajectory Prediction	4647	4	0.27
Store to LOCUS Persistence Schema	locus_geoshape	POI Detection	384	6	76.91
Read from LOCUS Persistence Schema	locus_geoshape	Path Detection	6982	1	0.27
Read from LOCUS Persistence Schema	locus_location_viavi	Path Detection	10000	5	0.33
Read from LOCUS Persistence Schema	uma_data_processed	Trajectory Prediction	4647	4	0.29

Read from LOCUS Persistence Schema	locus_geoshape	Path Detection	6982	1	0.39

As expected, write operations have the larger latencies which are expected due to the nature of Apache Hive software that uses the MapReduce framework to process queries [6], [7] .

2.2 Service latency model via LOCUS platform for Soft-Information-based localization

The soft information (SI)-based localization service has been implemented and deployed on the LOCUS platform. However, to properly evaluate the performance of the service, a model of the latency introduced by the SI-based localization service is needed. To this end, a latency model has been fit based on 5000 packages processed by the service. In particular, the latency was measured considering the time requested from when a measurement is available on the Clear_Data exchange to when the corresponding estimated position is available on the PositionAndNetworkInfo exchange on the LOCUS platform. The latency probability model is obtained fitting a Gaussian Mixture Model (GMM) to the measured latency data which results in

$$f(l) = 0.45 * N(208.20,358.56) + 0.1 * N(417.41,14211) + 0.45 * N(251.23,940.22)$$

where $N(\mu, \sigma^2)$ denotes a Gaussian probability distribution with mean μ and variance σ^2. Figure 15 shows the probability mass function (PMF) of the latency values and the fit GMM model. Figure 16 shows the empirical cumulative distribution function (ECDF) of the latency data. From the ECDF it can be observed that the latency is around 211ms at the 30th percentile, around 228ms at the 50th percentile, around 254ms at the 70th percentile, and around 308ms at the 90th percentile.

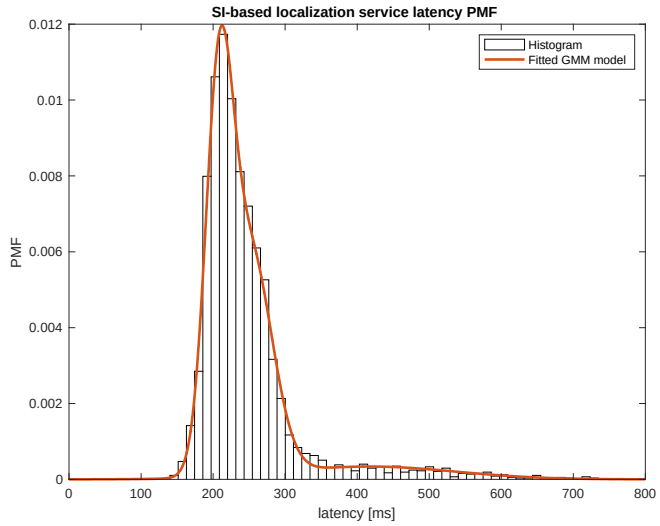


Figure 15: Latency PMF and fit GMM model

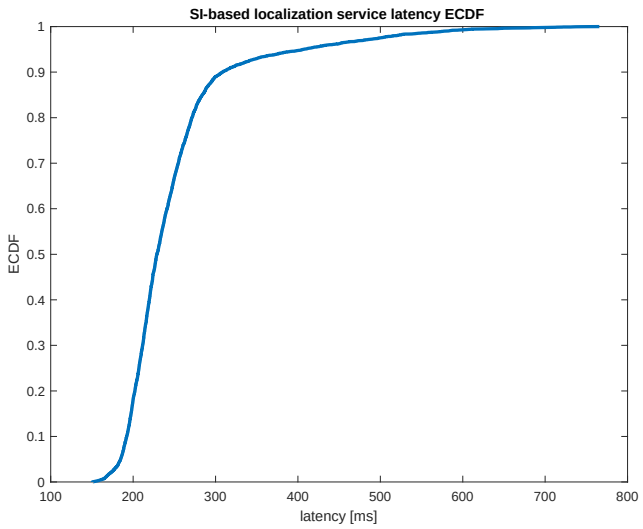


Figure 16: Latency ECDF

2.3 Location security and privacy

This subsection focuses on the anonymization module of the LOCUS system architecture. The anonymization module works as an interface module between the collection module and other blocks of the LOCUS architecture (see the LOCUS architecture). It acts to guarantee the privacy strength in the LOCUS platform.

We recall that there is a privacy threat whenever an adversary can associate the identity of a user to information that the user considers private. Privacy issues arise both in the acquisition of data, and in the subsequent release from the repositories where the data is stored. If we consider that an adversary may be able to obtain one or more transactions at the time they are acquired, a protection technique must be applied at each transaction.

The anonymization module has been designed to opportunely filter the platform input data and protect information which may reveal any details of personal information/activities about a specific user. Regarding the personal information, LOCUS puts more attention to the user's locations.

More specifically, the aim of the LOCUS anonymization module is twofold: first, it receives anonymization requests from other architecture modules and provide anonymized data. Second, it manages request to 3rd party servers, it can interact with Location Based Service (LBS) to provide privacy preserved users requests. LBS are location-aware services offered to the user from Location-based service providers (LSP).

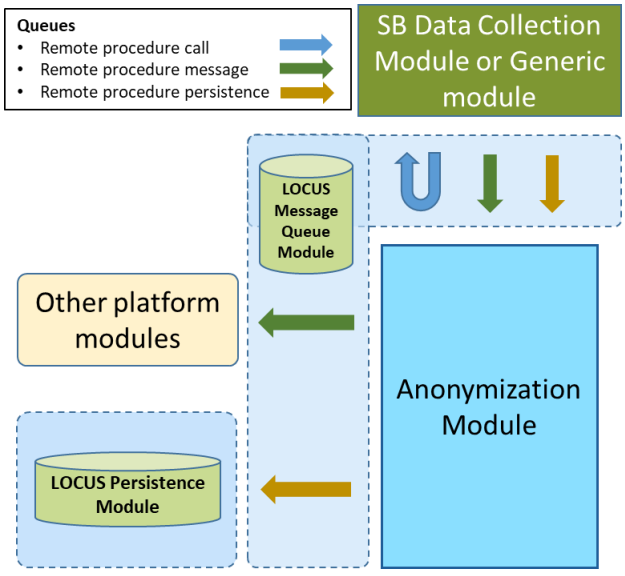


Figure 17: Architecture of the anonymization process

Figure 17 depicts the architecture of the privacy service and the interaction with the generic platform modules. The service is composed of single NFV Network Service interfaced by the LOCUS message queue module. Anonymization requests arrive from data collection module, afterwards, anonymized data can be routed by different queues.

As shown in figure, three queues are supported to enable multiple routing, they are: I) remote procedure call (topic name "LOCUS_rpc") to provide anonymized data directly to the entity that has generate the request; ii) remote procedure message (topic name is "LOCUS_msg") to forward the anonymized data on a different queue (the queue name is specified in the request); iii) remote

procedure persistence (topic name is “LOCUS_prs”) to store the anonymized data on the LOCUS persistent module.

The anonymization request supports geojson format, and different types of anonymization algorithms and levels. Four anonymization levels are used to define the set of fields that are involved in the anonymization process. Below how the definition of the anonymization level works.

> “LEVEL_0” : anonymization process is disabled.
> “LEVEL_1” : anonymization is applied only on the list of positions present in the request.
> “LEVEL_2” : anonymization is applied on the list of positions present and on the user identity.
> “LEVEL_3” : anonymization is applied on the list of positions, on the user identity, and on the timestamp of the request.

When the anonymization is enabled, 3 types of anonymizations can be selected, they are:

> “ROUNDING” : rounding is used to prevent exact matching with external data sources. It is used to reduce the level of detail in the data, regarding the positions, it works by removing decimal from float values. The effect is to report a subset of coordinates in a general point.
> “NOISE” : noising or masking, means adding random values to the original values of a data. Noise addition can prevent exact matching of continuous variables. The advantages of noise addition are that the noise is typically continuous with mean zero, and exact matching with external files will not be possible.
> “K-ANONYMITY“ : this anonymization type implements a data process to produce output where a specific position cannot be distinguished from at least the other k-1 positions in a group, more details about this anonymization type are provided below.

Finally, an additional field named “routingType” provides the selection of the response routing. Below, an example of the anonymization request where all the fields are present.

```
Geojson = {
“timestamp”: ‘2020-09-14 21:15:00’,
“userID”: ‘3C8226DE8770212B530A’,
“type”: “FeatureCollection”,
“name”: “dataName”,
“anonymizationLevel”: “LEVEL_2”,
“anonymizationType”: “rounding”,
“routingType”:”routingQueue”,
“features”:[ {
      “id”: “0”, “type”: “Feature”, “properties”: {“col1”: “name1”},
      “geometry”: {“type”: “Point”, “coordinates”: [22.237805348032513, 39.03284328258168]},
    },  ],
}
```

In the case of LBS, the identity and private information of a single user can be derived by the monitoring of multiple requests. The service contemplates the model where users execute queries with position and content requests to LBS. LBS uses individuals’ location data to provide their requested information, such as the nearest restaurant, or retail store. In this context, the service provider can identify places that these users frequently visit and reveal their personal information. To prevent these types of

attacks, the privacy service integrates k-anonymity and data aggregation functions. K-anonymity processes the data to make the output related to one user indistinguishable from at least other k-1 individuals. While result aggregation uses aggregate responses to respond to 3rd party, thus, the aggregate request is pushed to LBS. The privacy service uses kd-tree space data structures, it is a data structure that arranges geometric data for efficient search.

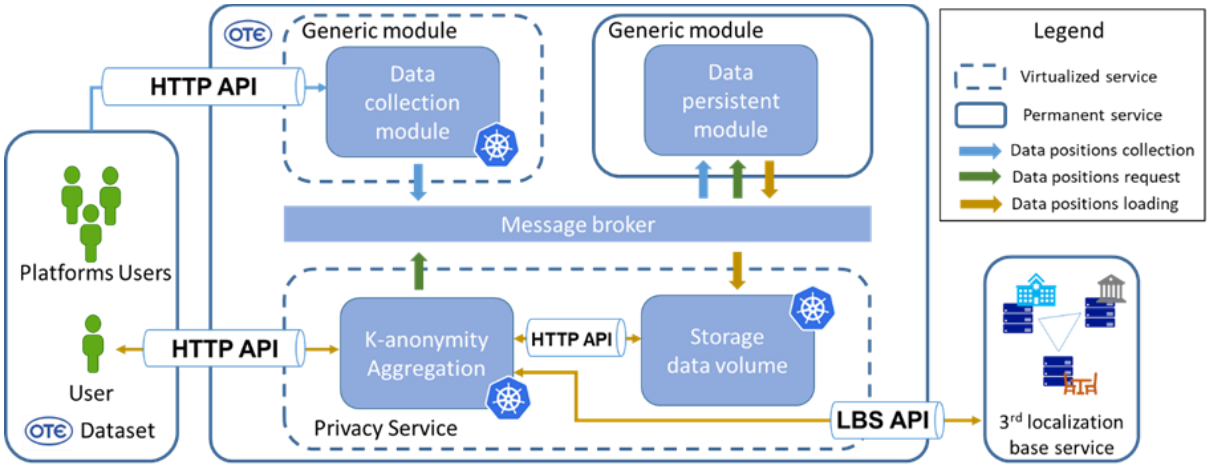


Figure 18: LOCUS privacy service

Figure 18 depicts the architecture of the privacy service and the interaction with the generic platform functions. The service is composed of two different blocks. The first one, the k-anonymity and aggregation function receives users' requests, apply the proposed algorithms, and forward the result to the LBS. The second block maintains the data structure used from the first function to apply the proposed algorithm.

Moreover, Figure 18 shows the data collection module and the data persistent module, both are generics platform modules, the first one allows the loading of user's data to the LOCUS Platform, and the second maintains the complete users' data available by other LOCUS functions and services functionalities. K-anonymity and aggregation function results can be shown on a generic map exposed via platform GUI

More in detail, the deployed service offers the following data flow.

For each user, the system performs a location update. Data are collected from the data collection module and stored in the data persistent module. HTTP APIs are used to connect datasets and data collection module.

Incoming requests arrive from the users to the privacy service. HTTP APIs are used to forward requests to the privacy service. The role of the k-anonymity and aggregation function is to filter the incoming user requests and to produce anonymous counterparts that can be safely forwarded to the LBS.

Starting from the user position, the privacy service pushes a request to the data persistent module via message broker bus, the request contains the current user position and the region size. All the user population positions that match the request region will be loaded into the storage data volume block. The data persistent module contains all population users' positions, but only a subpart of the data is suitable for the privacy service. An optimized data structure for position search is deployed on storage

data volume, multiple searches can be executed on storage data volume in an optimized way. The connection between the data persistent module and the privacy service data storage volume is recovered by the message broker bus.

To produce the anonymous counterpart, the privacy service works towards two main directions: I) removing any obvious identifiers that are part of the user request (e.g., ID, name); ii) the nearest neighbour users are searched on the kd-tree structure, and the anonymity is processed according to the privacy algorithm [8]. The main goal of the k-anonymity and aggregation function is to search and to select the nearest neighbour users that implement the anonymity set, it is the set that minimizes the probability to extract sensitive users' information. The entropy metric is used to measure the uncertainty of a group of users, the bigger the value of the entropy the more uncertain the group. Finally, the user group with the largest entropy is filtered and provided as output. The result is a cluster of data positions.

Request that includes the algorithm result is forwarded to LBS by proprietary LBS API. LBS result is processed from the privacy service to extract user suitable information. The outcome is sent to the user.

2.3.1 Integration with the LOCUS platform

The privacy service has been integrated in the LOCUS platform as part of the activities of WP6. The privacy service runs on LOCUS virtualization platform that is hosted in the OTE infrastructure. Data from use real-world mobility traces are connected to the platform. Data is provided from the OTE network operator; users are tracked through GPS coordinates with a resolution time of 15 minutes. Dataset is provided after an anonymization process and the dataset covered an interval time of 24 hours in September 2020, a complete description of the dataset is reported in D5.3 [4].

The location data records GPS coordinates together with others network information. The dataset used has over 20 000 000 location traces splitting into 3 different location areas in Greece.

The first version of the code is available in the project repository, it provides a platform running function image that exposes RESTful HTTP APIs to enforce privacy functionalities to the platform. The interface module has been developed in python by using Flask library, three sets of RESTful HTTP APIs are been considered: i) login API; ii) management API; and iii) service API. Platform users can establish a connection to the service by using login API, update location position by using management API, and submit privacy preserved requests to external Location Base Service by using service API. K-anonymity requests can be served from primary RESTful HTTP APIs, as well as from a dedicated queue in the RabbitMQ message broker that implements communication through an AMQP protocol.

We integrate the privacy functionalities on PoC#1, and PoC#3. Specifically, a Graphical User Interface (GUI) map has been developed to show the user positions along a moving path. For each position, a k-anonymity privacy request is generated, and the relative extracted k-anonymity group is shown on the map. The GUI map has been developed on top of the REACT OpenGL library and it is connected to the platform message broker by a consumer module and a WebSocket protocol.

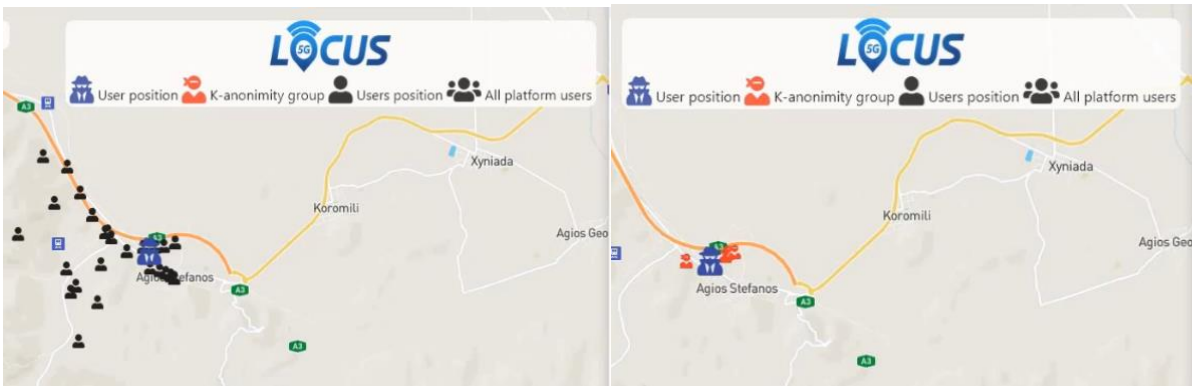


Figure 19: Privacy demo GUI

2.4 Localization enablers

Indoors spaces are characterized by harsh conditions for radio propagation due to signal blocking by walls, multipath, and clutter. This requires the capability of learning the environmental conditions as well as fusing heterogeneous measurements.

2.4.1 5G Localization Based on Soft Information

Soft information (SI)-based localization for 5G networks has been developed in the context of WP3 with research results reported in deliverables D3.1, D3.3, D3.4 and D3.7 ([9]– [13]). Moreover, a SI-based localization service has been implemented in the context of the POC#1 as reported in deliverables D4.4 [1] , D5.4 [2] and D6.2 [14]. SI-based localization leverages machine learning techniques to provide a statistical characterization of the relationship between measurements, contextual information, and user equipment position [15]. Such approach overcomes the limitations of conventional localization algorithms, providing enhanced location awareness in 5G an beyond wireless networks [16]. Moreover, SI-based approach is inherently capable of fusing data measurements from heterogeneous technologies (e.g., LTE, 5G, Wi-Fi, and UWB)

2.4.2 Fusion based positioning

Indoors spaces are characterized by harsh conditions for radio propagation due to signal blocking by walls and clutter. For a point to have more than two reference points visible, the density of a deployment for location should normally be very high and costly. On the other hand, in indoors spaces it is common that more than one radio technology is deployed to serve different purposes: Wi-Fi and 4/5G femtocells for communications, UWB for localization, etc. While these signals are subject to the aforementioned challenges for location, the ability of using signals from different technologies to obtain ranges for trilateration greatly increases the chances of a target being in range of at least three reference points.
With opportunistic fusion, this ability is added to the network, where several technologies are used to estimate distances to a single target, and then use those distances to estimate the location. The first approach is described in D3.3 [11], where simulations of UWB+mobile are shown. The idea is furthermore refined in D3.4 [12], where a weighting mechanism is added to give more relevance to more precise ranges.

2.4.3 Device-free Localization

In several applications and environments, it is important to track people, objects, and vehicles that are not exchanging measurements with the network for localization purposes. Such localization is called device-free localization [17]. Theoretical foundations, multi-target tracking algorithms, and network experimentations on device-free localization and tracking have been conducted and presented in deliverables D3.5 [18], D3.6 [19] and D3.8 [20].

3 Deployment, Proofs-of-Concept, and Example Results

In this section we examine the set of PoCs developed throughout the project and integrated with the LOCUS Platform. We explain the various storylines demonstrated by the PoCs and showcase sample results.

3.1 Proof of Concept 1: Network management based on location information

3.1.1 Storyline 1: Building radio maps for coverage hole detection and correction

The storyline is about coverage optimization. It consists of predicting Radio Environment Maps (REM), then leveraging the predicted REM to detect a coverage issue and eventually propose corrective measures to optimize coverage. More details about the use case description can be found in D4.1 [21].

3.1.1.1 Concept

Reference Signal Received Power (RSRP) levels (e.g. as measured during drive tests) can be predicted all over an area using interpolation techniques. The technique employed within this storyline is the Kriging. Such prediction allows for early problem detection such as coverage holes, where RSRP levels are below a given threshold. Thus, a proactive intervention for the correction is made. In this storyline, coverage holes are detected after RSRP maps are predicted. A matrix of power correction is provided to compensate low RSRP levels. For more details, explanations are provided in D4.1 [21]. The solution is deployed as cloud native solution within the LOCUS platform and thus opens the door for Self-Organized Networks (SON), where networks problems are automatically detected and corrected.

3.1.1.2 End-to-end execution

In [14] an overall description of the storyline is provided. However, the service details with the different functions that compose it are described within this document.
The coverage optimization is deployed “as a service” and consists of three different functions running each over docker containers which are pipelined together and exchanging messages via the RabbitMQ bus. It is worth mentioning that the service has been packed using the juju charms operator [22] supported by the LOCUS MANO (and ETSI OSM). More details about the “dockerization” and the “packaging” processes with juju charms can be found in D4.4 [1] .
A first function deemed “Prediction”, uses kriging interpolation, to predict radio coverage maps based on real time measurements which are collected from UMA network, by a moving User Equipment (UE). It is worth noting that measurements can be geo-tagged with any available positioning service, by default this function relies on position estimates from the “soft information”. The latter publishes the results on “PositionAndNetworkInfo” topic. The service examines the measurements and extracts the collected coordinates (x, y) along with the associated RSRP level and the serving cell. Based on this, the “Kriging” function applies the Kriging algorithm for interpolation, and prediction RSRP measurements on coordinates not covered by UE movement. The result of this service is a matrix containing the predicted RSRP level on a new point with its associated serving cell. The matrix is then sent via the RabbitMQ bus on “krigingout” exchange. The data shown in Table 4 is an example of coverage

prediction results using one soft-information positioning, these positioning estimates are generated based solely on LTE measurements. The first two columns of the table represent the coordinates of points where RSRP prediction is made. The third and fourth columns are the predicted RSRP levels. We note that predictions can be made based on multiple localization techniques. The last column represents the PCI of the cell that serves the corresponding coordinates.

Table 4: Example Interpolation results published to RabbitMQ

X	Y	RSRP	PCI
-4.14352	5.96559	-92.6606	234
-21.1386	35.5634	-91.5973	234
-10.8735	13.218	-96.3898	234
2.98401	-6.34011	-91.983	221
-11.7565	30.3197	-93.8928	221
-11.4536	-13.2449	-91.6269	221
-20.7711	-21.8721	-91.2173	221
-23.6753	4.14155	-95.0508	234
-10.1683	7.34157	-95.6198	234
0.817761	36.6388	-93.5353	234
-18.7371	-23.6031	-91.2173	234
-23.8949	26.4135	-93.0206	234
-1.06023	-20.1202	-91.9442	234
-5.31834	12.363	-93.815	234
-16.6461	6.12231	-96.739	221
-12.1231	22.5976	-95.5969	234
6.28226	-16.7464	-93.3038	221
-2.4886	4.07964	-91.6471	234
-9.158	12.1137	-95.6581	234
-2.70409	20.2112	-92.473	234

The second function deemed “Detection” relies on the input of the “Prediction” function, and examines the matrix sent via the RabbitMQ bus to detect any coverage holes. Particularly, this service examines all the predicted RSRP levels in different points and extracts the ones that show values less than a pre-fixed threshold along with their serving cells. The result of this service is a matrix containing all the detected points with associated RSRP levels less than the threshold and their serving cells. This function publishes its findings on the RabbitMQ bus using the “covholes” exchange. The data in Table 5 represents an example of the one sent via the RabbitMQ bus. It shows the coordinates of all the different points having RSRP levels less than the pre-fixed threshold. The first two columns represent the coordinates of such points. The third one is the level of RSRP and the fourth is the Physical Cell ID (PCI) of the serving cell.

Table 5: Example Coverage Holes detected and published on RabbitMQ

X	Y	RSRP	PCI
-23.6753	4.14155	-95.0508	234
-10.1683	7.34157	-95.6198	234
-9.158	12.1137	-95.6581	234
6.28226	-16.7464	-93.3038	221

-10.8735	13.218	-96.3898	234

The third function is called “Correction” and is used to calculate the level of power (ΔP) to be adjusted for the cells that has shown RSRP levels less than the threshold. The result of this function is a matrix containing the different cell IDs (*i.e.*; PCI) with the associated level of power (ΔP) to be adjusted. The data in Table 6 is an example of the output of the function. The first column represents the PCI and the second is ΔP.

Table 6: Example Power Deltas to eliminate coverage holes

PCI	delta P
221	2.947
234	3.389
280	0.0

Figure 20 is an illustration of the service with its different functions composing it. The figure shows how the functions are pipelined together and are exchanging messages via the RabbitMQ bus.

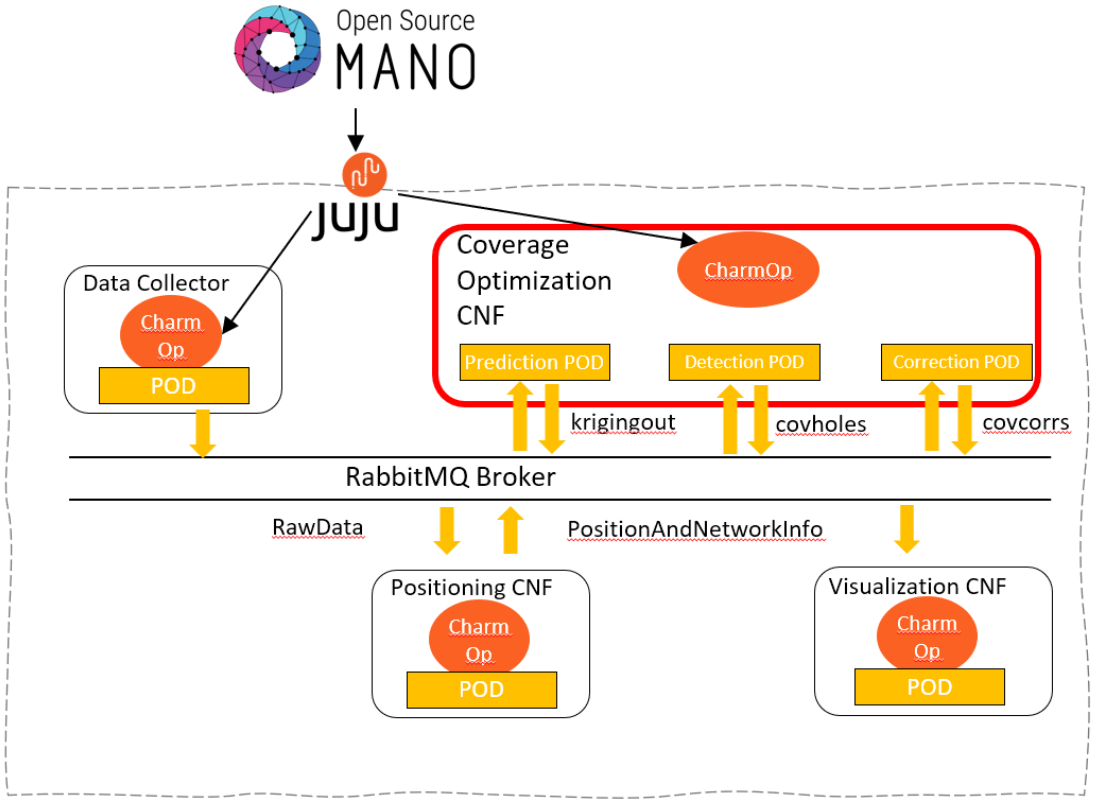


Figure 20: Coverage Optimization Service Flow

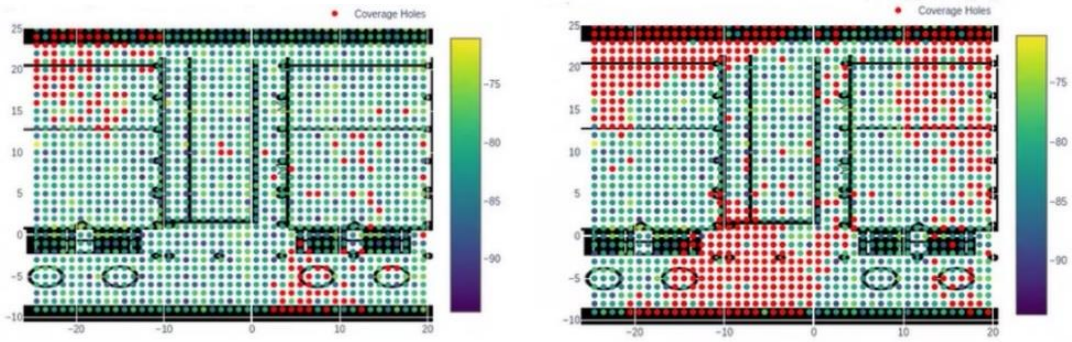
3.1.1.3 Mapping on the platform

The service is deployed using several functional blocks from the LOCUS platform. Below is a list of all the functional blocks composing the service. More details about such functional blocks can be found in D2.5 [23]. However, in this document we describe how the different service functions are using those blocks to run on the platform:

- **LOCUS Persistence:** The prediction function can be "warmed up" with a pre-existing dataset, store in the persistence module, to which real-time data is appended before executing the kriging interpolation.
- **Message Queue**: This block is used to receive measurements from the measurements and position estimates in real-time from other deployed services. For example, the predicted RSRP matrix, the matrix of points with coverage issues and the matrix of level of power to be adjusted for the different cells are published and exchanged via the broker as previously explained.
- **Privacy Module**: This function is deployed to provide an additional source of positioning estimates, that are generated by anonymizing, using different techniques, the positioning estimates generated by other services. As far as the storyline is concerned the coverage optimization network function can be configured to consume from this anonymization positioning source as any other positioning source.
- **LOCUS Virtualization Management and Orchestration (MANO):** The LOCUS MANO function is used for automatic configuration, orchestration, and operation of the different Virtualized Network functions (VNFs). More details about MANO can be found in D4.4[1] and D4.3[24].
- **LOCUS API layer and platform control:** The API catalogue (with its service subscription capabilities) together with the Analytics Coordinator allow platform users to automate the deployment of the coverage optimization analytics service, leveraging on the MANO capabilities above.

3.1.1.4 Results

First, the storyline is used to examine how the different localization techniques affect the predicted RSRP maps, through their varying positioning accuracies. Relying on the different results sent by the "soft information" service, the different RSRP maps are displayed and compared with the ground-truth to examine several localization estimation techniques. In fact, the "soft information" service relies on different technologies such as LTE, UWB, Wi-Fi or any combination of those to perform predictions. For example, in the figures below we have different coverage maps. The first is based on the localization technique relying on LTE technology, while the second on UWB and Wi-Fi. We notice the difference between the two maps. For example, the first displays more coverage holes (illustrated in red points) than the second. This shows how



different localization techniques affect the predictions. It is worth to be noting that this demonstration is to be performed offline and not with real time collection.

The second story is used for examining the accuracy of the Kriging algorithm. This demonstration will be performed is real time. In particular, the error evolution of the Kriging while collecting the

measurements in real time will be shown. The purpose will be to show the fast decrease of the error while measurements are collected, and thus highlighting the efficiency of the Kriging algorithm. (a result can be added here)

With the ability to estimate power compensation to eliminate coverage holes. It is possible to interact with the UMA deployed LTE nodes to change the transmission parameters to implement this compensation, it consists of adjusting the power of cells that are showing coverage issues.

3.1.2 Storyline 2: Contextualized indicators for diagnosis and troubleshooting

3.1.2.1 Concept

This storyline aims to prove the benefits of generating contextualized indicators basing on localization and cell distribution around the scenario. Information from context (e.g. weather, traffic, social events, etc.) is gaining importance for the network management paradigm. Nonetheless, this information is not always available, and it is usually more difficult to collect it. Here, contextualized indicators are computing using users' position and radio measurements as input. This enables the opportunity of generating and updating them very fluently. Detailed explanations are present in D4.1 [21].
Then, these indicators are sent to the LOCUS Platform in order to exploit them for failure detection. For this part, ML-aided algorithms allow to determine whether any of the cells in the scenario is down or, conversely, all the cells are working as expected. For more details, they are also included in D4.1 [21].

3.1.2.2 End-to-end execution

This subsection describes the functionality details about the implementation on the LOCUS Platform. The functionality is defined as an analytics service and deployed as a VNF based on Docker through the LOCUS MANO and ETSI OSM.
The communication is done via a RabbitMQ bus through different topics. Figure 21 illustrates the blocks that are needed for the service operation, as well as the RabbitMQ topics in use. Firstly, WP3 output represent the positioning information together with radio information (PositionAndNetworkInfo) that is used as input for the storyline. Then, the contextInd-funct block is in charge of computing the areas and associating each measurement to its corresponding area. It also updates the indicators of each area with each new measurement. The storyline is based on RSRP measurements, but it can be directly extrapolated to use any other available radio metric. The contextualized indicators are sent via rsrpAllAreas_topic to a block that determines the current status of the network scenario. The output of this block (Diagnosis_topic) together with additional information generated in the contextInd-funct block are used for the representation in the User Interface (UI).

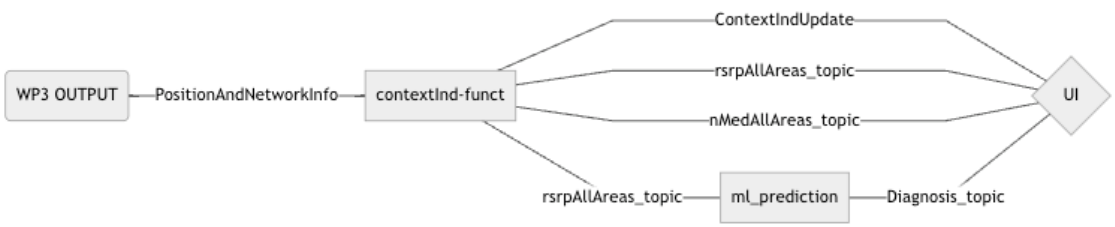


Figure 21: Contextualized Indicators Service

Regarding the Diagnosis topic, available datasets were processed in order to generate synthetic datasets with different conditions. The datasets provided by WP3 output contained positioning information together with radio information. It contained serving and neighbouring cells for each sample. In order to emulate a cell outage, specific cell should not appear at all on the whole dataset. Here, we processed the dataset while dropping the information from specific Physical Cell ID (PCI) and considering the first neighbouring cell as serving cell in those case where the specified PCI corresponded with the serving cell. Number of sites was decrease by one in samples where the PCI was present. After repeating the process for each PCI of interest, it was possible to train the ML models that then predict the given situation. This approach may not be very realistic in terms of interference, but it has advantages in order to be applied on commercial networks, where cells cannot be turned off for generating datasets.

3.1.2.3 Mapping on the platform

The service is deployed using several functional blocks from the LOCUS platform. Below is a list of all the functional blocks composing the service. More details about such functional blocks can be found in D2.5 [23]. However, in this document we describe how the different service functions are using those blocks to run on the platform:

Message Queue: This block is used to receive measurements from the measurements and position estimates in real-time from other deployed services.

Privacy Module: This function is deployed to provide an additional source of positioning estimates, that are generated by anonymizing, using different techniques, the positioning estimates generated by other services.

LOCUS Virtualization Management and Orchestration (MANO): The LOCUS MANO function is used for automatic configuration, orchestration, and operation of the different Virtualized Network functions (VNFs). More details about MANO can be found in D4.4 [1] and D4.3 [24].

LOCUS API layer and platform control: The API catalogue (with its service subscription capabilities) together with the Analytics Coordinator allow platform users to automate the deployment of the coverage optimization analytics service, leveraging on the MANO capabilities above.

Table 7: JSON format example of "PositionAndNetworkInfo" topic.


```
{ "info":{
    "BS":[
      [
        -20.39,
        13.9
      ], [
        -11.73,
        -0.5
      ], [
        -18.34,
        -0.5
      ], [
        13.1,
        13.9
      ], [
        -15.3,
        -0.5
      ], [
        -18.89,
        13.9
      ], [
        7.05,
        13.9
      ], [
        5.0,
        -0.5
      ]
    ],
    "distances":[
      11.198,
      11.685,
      14.35,
      27.227,
      15.525,
      25.626,
      16.5,
      19.228
    ],
    "ID":[
      "D7A6",
      "8F2B",
      "903B",
      "CF0F",
      "3c:28:6d:b2:c9:1f",
      "08:b4:b1:70:4b:65",
      "3c:28:6d:b2:e2:b0",
      "3c:28:6d:b2:e2:0b"
    ],
    "BS_LTE":[
      [
        -19.09,
        3.87
      ]
    ],
    "distances_LTE":[
      11.465
    ],
    "ID_LTE":[
      "280"
    ],
    "bsLte":{
      "221":{
        "X":6.12,
        "Y":3.9,
        "Z":2.45,
        "Frequency":2.85,
        "Ptx[dBm]":-6.8
      },
      "234":{
        "X":12.4,
        "Y":11.99,
        "Z":3.5,
        "Frequency":2.85,
        "Ptx[dBm]":-6.8
      },
      "232":{
        "X":5.94,
        "Y":15.31,
        "Z":3.5,
        "Frequency":2.85,
        "Ptx[dBm]":-6.8
      },
      "271":{
        "X":-13.05,
        "Y":11.76,
        "Z":3.5,
        "Frequency":2.85,
        "Ptx[dBm]":-6.8
      },
      "280":{
        "X":-19.09,
        "Y":3.87,
        "Z":2.45,
        "Frequency":2.85,
        "Ptx[dBm]":-6.8
      },
      "274":{
        "X":17.3,
        "Y":-6.3,
        "Z":10.869,
        "Frequency":2.85,
        "Ptx[dBm]":-6.8
      },
      "227":{
        "X":17.3,
        "Y":1.6,
        "Z":10.869,
        "Frequency":2.85,
        "Ptx[dBm]":-6.8
      },
      "296":{
        "X":17.3,
        "Y":19.04,
        "Z":10.869,
        "Frequency":2.85,
        "Ptx[dBm]":-6.8
      },
      "213":{
        "X":17.3,
        "Y":-6.3,
        "Z":6.863,
        "Frequency":2.85,
        "Ptx[dBm]":-6.8
      }
    },
    "bsUwb":{
      "C215":{
        "X":8.0,
        "Y":-0.5,
        "Z":2.0
      },
      "0088":{
        "X":4.3,
        "Y":13.9,
        "Z":2.0
      },
      "9105":{
        "X":11.3,
        "Y":-0.9,
        "Z":2.0
      },
      "CF0F":{
        "X":13.1,
        "Y":13.9,
        "Z":2.0
      },
      "D10C":{
        "X":3.45,
        "Y":14.35,
        "Z":2.0
      },
      "5720":{
        "X":13.42,
        "Y":14.35,
        "56B9":{
          "X":3.3,
          "Y":22.91,
          "Z":2.0
        },
        "8F2B":{
          "X":-11.73,
          "Y":-0.5,
          "Z":2.0
        },
        "D7A6":{
          "X":-20.39,
          "Y":13.9,
          "Z":2.0
        },
        "903B":{
          "X":-18.34,
          "Y":-0.5,
          "Z":2.0
        },
        "45B2":{
          "X":-12.03,
          "Y":13.9,
          "Z":2.0
        }
      },
      "bsWiFi":{
        "3c:28:6d:b2:e2:0b":{
          "X":5.0,
          "Y":-0.5,
          "Z":2.0
        },
        "08:b4:b1:70:4b:65":{
          "X":-18.89,
          "Y":13.9,
          "Z":2.0
        },
        "3c:28:6d:b2:e2:b0":{
          "X":7.05,
          "Y":13.9,
          "Z":2.0
        },
        "08:b4:b1:70:44:7a":{
          "X":10.05,
          "Y":14.35,
          "Z":2.0
        },
        "08:b4:b1:70:47:df":{
          "X":3.3,
          "Y":22.91,
          "Z":2.0
        },
        "3c:28:6d:b2:c9:1f":{
          "X":-15.3,
          "Y":-0.5,
          "Z":2.0
        }
      },
      "UWB_Info":{
        "D7A6":[
          "Pos:[-20.39, 13.9]",
          "Distance:11.198",
          "Quality:64"
        ],
        "8F2B":[
          "Pos:[-11.73, -0.5]",
          "Distance:11.685",
          "Quality:64"
        ],
        "903B":[
          "Pos:[-18.34, -0.5]",
          "Distance:14.35",
          "Quality:64"
        ],
        "WiFi_RSSI":{
          "3c:28:6d:b2:c9:1f":[
            "Pos:[-15.3, -0.5]",
            "Distance:15.525",
            "RSSI:-74"
          ],
          "08:b4:b1:70:4b:65":[
            "Pos:[-18.89, 13.9]",
            "Distance:25.626",
            "RSSI:-94"
          ],
          "3c:28:6d:b2:e2:b0":[
            "Pos:[7.05, 13.9]",
            "Distance:16.5",
            "RSSI:-88"
          ],
          "3c:28:6d:b2:e2:0b":[
            "Pos:[5.0, -0.5]",
            "Distance:19.228",
            "RSSI:-102"
          ]
        },
        "Lte_Info":{
          "LteCqi":"2147483647",
          "LteLevel":"4",
          "LteRSRP":"-73",
          "LteRSRQ":"-7",
          "LteRSSI":"-59",
          "LteRSSNR":"244",

"LteTimingAdvance":"2147483647",
          "LteDbm":"-73",
          "LteAsuLevel":"67"
        },
        "neighbour_Lte":{
          "NumberOfSites":"1",
          "Site0":{
            "MNC":"05",

"MobileNetworkOperatorID":"21405",
            "RSRQ":"-6",
            "CI":"34438",
            "Band":"7",
            "MCC":"214",
            "Dbm":"-76",
            "LevelSignalStrength":"4",
            "RSSI":"-51",
            "EarFCN":"2850",
            "PCI":"280",
            "TAC":"1",
            "Bandwidth":"20000",
            "PLMN":"12F450",
            "eNB":"134",
            "RSRP":"-76",

"MobileNetworkOperatorName":"LTE
UMA",
            "CQI":"2147483647",
            "RSSNR":"2147483647",
            "Registered":"true"
          }
        },
        "NR_Info":{

        },

        "id_mobile":"1",
        "Timestamp":"2022-03-
09T09:52:47.400",
        "position":[
          -6.93137569222067,
          8.506071995453507
        ],
        "position_LTE":{
```

	"Z":2.0 },	], "CF0F":["Pos:[13.1, 13.9]", "Distance:27.227", "Quality:64"] },	} } }

3.1.2.4 Results

This service has proved to work within the proposed scenario composed by 5 radio cells. It has been configured with a 20 samples windows, which means that it can detect a cell outage after 20 received samples. This can be adjusted meaning that increasing this number accuracy will be improved but reactiveness will be reduced.

The service works in the scenario described in D6.2 [14] where UEs are moving around the scenario and reporting the information in real time. Figure 22 represents the UI Map as result of the movement of several users during a period of time.

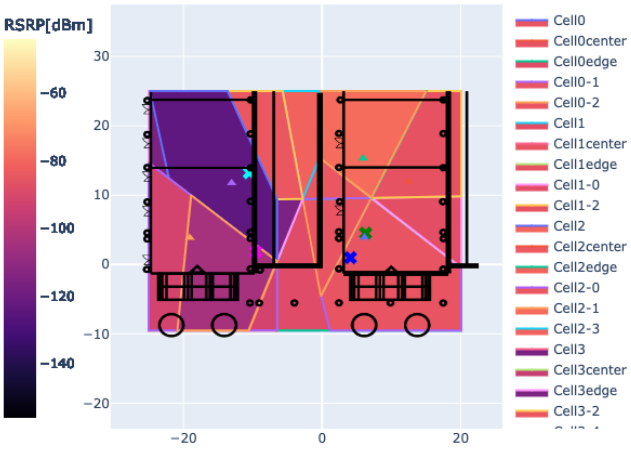


Figure 22: Contextualized Indicator Map

3.1.3 Storyline 3: Network Planning and Redesign Using Hybrid Spatial Network Clustering

3.1.3.1 Concept

The concept to be proved in this storyline is that network services in urban areas can be insufficiently managed due to the varied density of mobile UEs in different subareas. An unsupervised method that clusters together regions considering both mobility and network- related characteristics can indicate these issues and provide useful insights to enable network optimization actions, and specifically azimuth steering of cells with inadequate performance.

3.1.3.2 End-to-end execution

As part of WP4 work, the concept of azimuth steering based on clustering results has been presented in D4.5 [25]. The following figure (Figure 23) displays the analytics services deployed for the end-to-end execution of this storyline. In more detail, the end-to-end execution pipeline of this service consists of two distinct analytics services, specifically a function/service (as in previous storylines) is assumed to deliver UE position information in the Persistence module, then a hybrid spatial-network clustering

service is instantiated, and its results are stored and then considered as input for the azimuth steering service. A more detailed description follows:

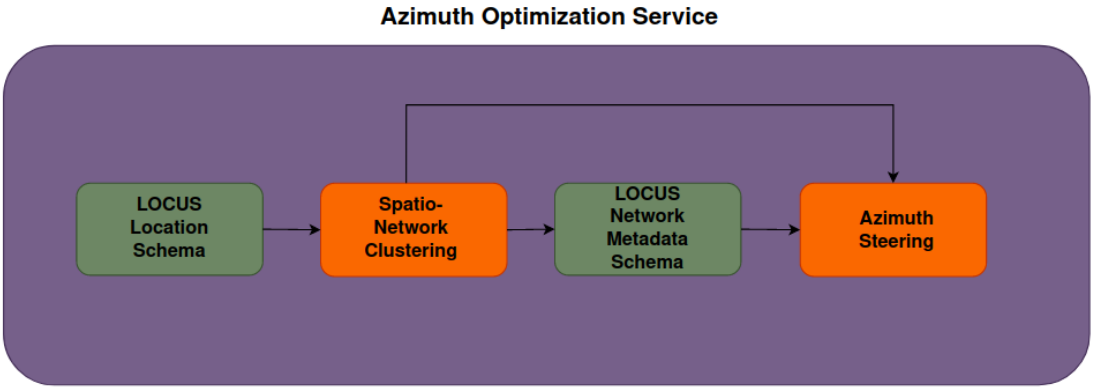


Figure 23: Network azimuth optimization analytics service pipeline – PoC#1

Network Spatial Clustering

Each UE position is assigned to a spatial bin. The input space is thus rasterized, making the clustering process faster and more scalable.

Each measurement is also assigned to a network KPI bin (Low/Medium/Good). These bins will serve as classes that will be assigned to the discovered clusters with a majority rule. Cluster discovery is an iterative process that detects large concave areas in the grid labelled as Bad with smaller convex areas labelled as good. If a ratio condition is satisfied, the internal area is re-labelled as Bad, thus forming a larger convex Bad region, which will form a Cluster. In the end of this process, the N largest Clusters of each category will have been discovered.

Azimuth Steering

The focus on this part is concentrated on the bad clusters. For each cell, we assign a Bad/Medium/Good service label based on the number of clusters of each class that it is serving. For every "Bad" labelled cell, we calculate the centre of gravity (this is geometric centre weighted by the clusters population) of each serving clusters and fix its orientation to face that centre.

3.1.3.3 Mapping on the platform

The functional blocks composing this service include:

- **The API Catalogue** to offering the presented analytics service to the various platform users, and trigger its automated activation in the platform.
- **The Analytics Coordinator** for managing the analytics service decomposition into individual virtualized analytics functions implemented as one or more NFV Network Services (to be then orchestrated by the LOCUS MANO).
- **The LOCUS Virtualization Management and Orchestration** (MANO) which is used for automatic configuration, orchestration, and operation of the different functions of the service.
- **The LOCUS SDK** and specifically the "Model I/O" module is used for communication with the LOCUS Persistence Module (data read/write).

- **The LOCUS Persistence Module** that persists both incoming data used by the various services, but also the results of the of those services.
- **The Data Operations Controller** (based on Apache Airflow [26]), which is responsible for orchestrating the pipeline.
- **The Service Discovery Module** (based on Consul [3]) that constitutes the service registry.
- **The LOCUS API Gateway**, responsible for the exposure of this service to external 3rd parties.

A UI for the purposes of demonstration complements the aforementioned components, acting as a 3rd party application.

3.1.3.4 Results

The experiments indicate that this pipeline can effectively discover insufficiently served Clusters of mobiles and exploit their spatial and network features to re-configure cells azimuths automatically. Figure 24 presents the result of the clustering service.

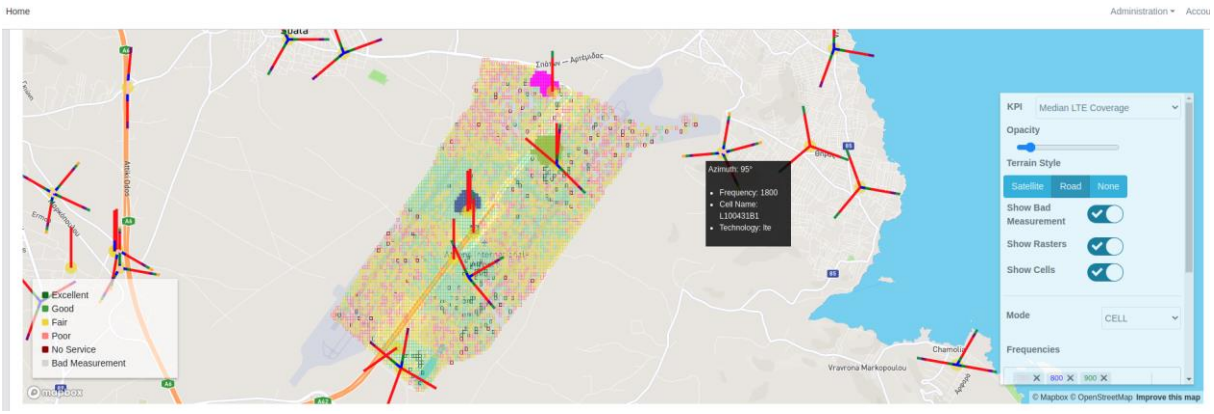


Figure 24: Indicative view of the hybrid clustering results

3.2 Proof of Concept 2: Network-assisted self-driving objects

3.2.1 Storyline 1: Logistics in a seaport terminal using Automated Guided Vehicles (AGV)

The aim of PoC#2 is to evaluate the feasibility of introducing accurate navigation solutions based on 5G technology for Automated Guided Vehicles (AGVs) operating in a seaport arena. The behaviour of a simulated AGV in port logistic scenarios are analysed to evaluate the accuracy required by the different cases and the sensitivity to the 5G positioning system accuracy tolerance.

3.2.1.1 Integration of PoC#2 simulation environment with LOCUS platform

A simulator of a relevant logistics use case based on the usage of AGVs has been implemented as in Figure 25 (see D6.1 [27], D6.2 [14] for a detailed description and explanation of the architecture blocks) and used to evaluate the performances in terms of positioning accuracy that can be reached using the 5G positioning system previously defined to validate the effects that errors affecting the position measurements can affect an AGV fully driven using such technology. The simulated environment can also consider the latency introduced by the LOCUS platform, and how it may affect the AGV navigation.

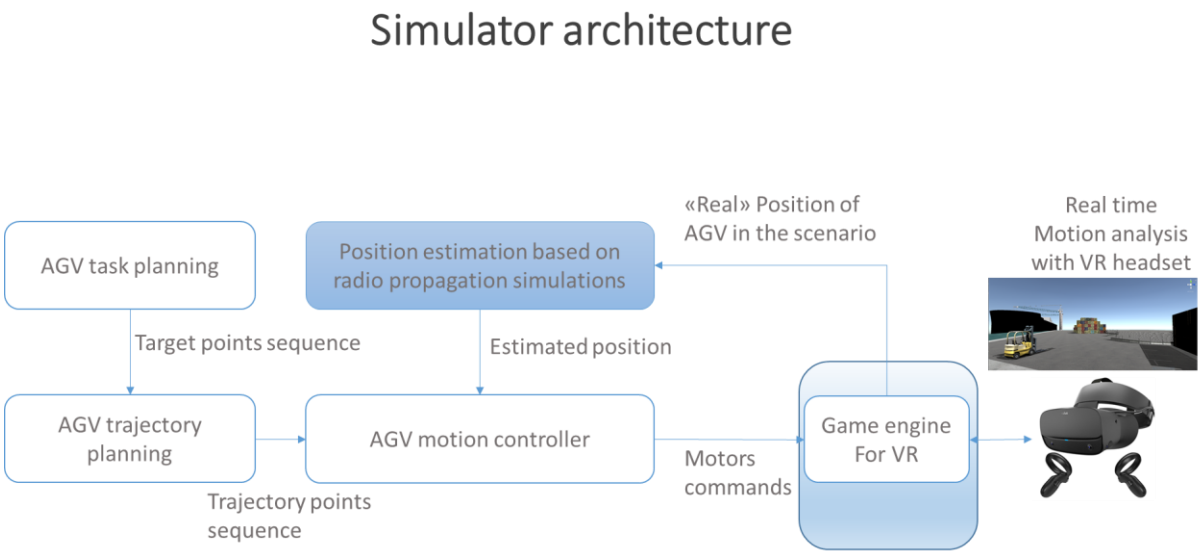


Figure 25: Architecture of PoC#2 simulator

3.2.1.2 Execution

To achieve the previous goal, PoC#2 utilize the localization enabler (LEN) based on the soft information approach developed in WP3 (see D3.7 [13]), that has been integrated in the LOCUS platform and operated in PoC#1. In the architecture of the PoC simulator (Figure 25), this is represented by the blue block “Position estimation based on radio propagation simulations”.

3.2.1.2.1 Generation of port map and trajectories

The map of the port arena has been modelled in a json file to be imported in the radio propagation simulator. A Java program has been written on purpose to generate massively trajectories considering position of freight and loading crane. The set of generated trajectories are those expected to be used by the AGVs when moving freight from the loading area to the stocking piles and vice versa.

The model included 40 loading locations x 3 stocking corridors for a total of 120 trajectories. They considered real AGV acceleration, deceleration, and steering radius. Generated datasets (recorded in a json file) consist of x-y coordinates, x-y speed components, taken with a time interval of 0.1 s.

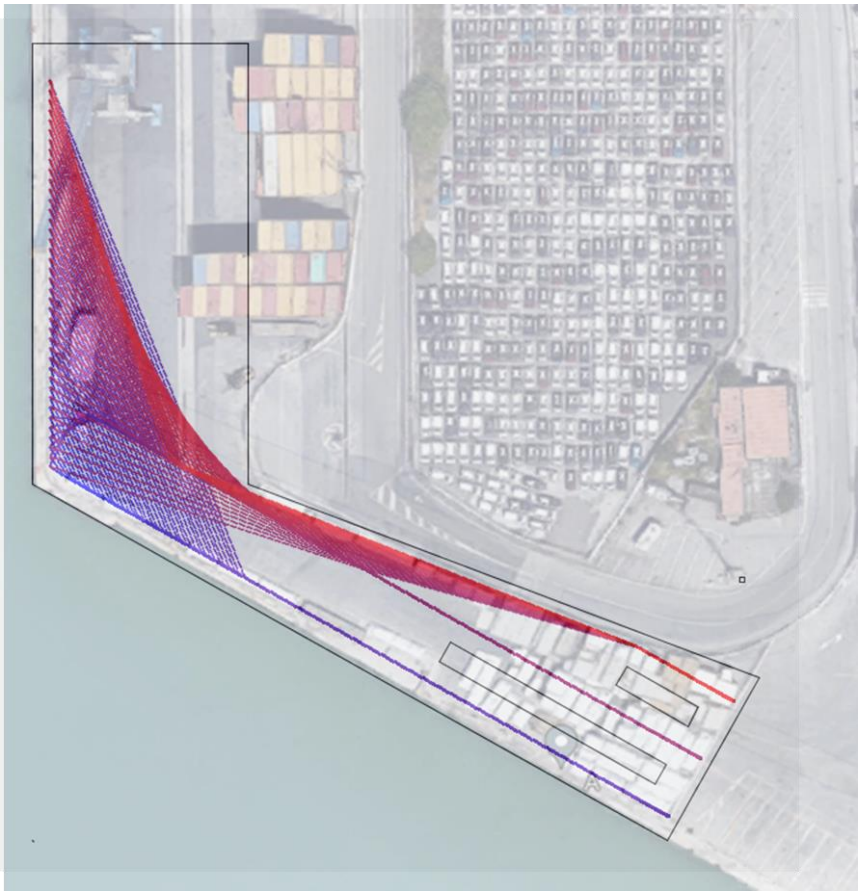


Figure 26: Overall set of generated trajectories shown in the port arena

3.2.1.2.2 Localization Enabler in port area simulation

The localization enabler based on the soft information (SI) approach that was developed in WP3 [15], deployed on the LOCUS platform, and tested in the context of the PoC#1 has been exploited to provide localization uncertainty models for the PoC#2. In particular, the 5G-NR localization uncertainty models have been determined in a port area with various gNBs configurations. The uncertainty models are obtained through 3GPP-compliant simulations in 3GPP standardized scenarios (in the absence of port areas specifications, settings with characteristics similar to real port areas have been used). Specifically, the wireless environment in areas without obstacles are modelled as rural macro (RMa) scenarios while the areas with containers (metal obstacles) are modelled as indoor factory SH (InF-SH) scenarios, both with spatial consistency. The areas in line-of-sight (LOS) and non-line-of-sight (NLOS) for each gNB are determined geometrically based on the container's placement.

Figure 27: Localization error heatmap for different gNBs deployments. The coordinates on the axis and the magnitude of the localization error are in meters

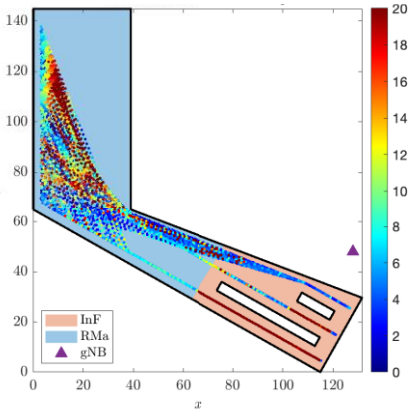


Figure 27a: Single gNB

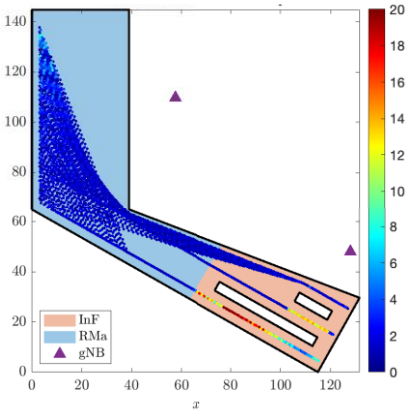


Figure 27b: Two gNBs

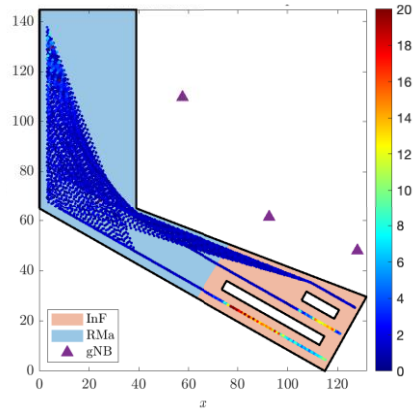


Figure 27c: Three gNBs

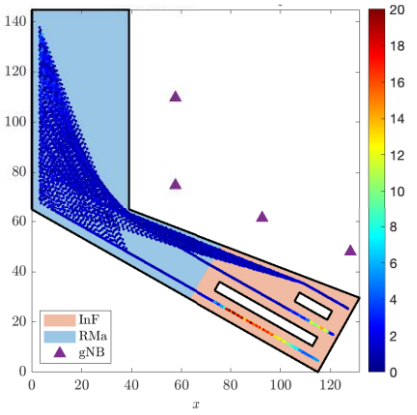


Figure 27d: Four gNBs

Localization is performed through the downlink transmission of the 5G-NR positioning reference signal (PRS) with a carrier frequency of 4GHz and a bandwidth of 100MHz. The PRS is then processed to extract time-of-arrival (TOA) and angle-of-departure (AOD) measurements, which are both exploited (fused) by SI-based localization method to infer the position of a moving automatic guided vehicle (AGV). The localization performance and error models are obtained based on 120 realistic simulated AGV trajectories. Figure 27 shows the localization error heatmap for the trajectory samples in the port area with different gNBs configurations.

The localization uncertainty models are obtained by fitting a 2-dimensional multivariate Gaussian distribution to the localization error for the different gNBs deployment configurations. Such models approximate the joint probability distribution of the localization error on the x and y coordinates.

The uncertainty models are obtained for the two areas (RMa and InF-SH) based on the number of deployed gNBs. Table 7 and Table 8 show the localization uncertainty models considering the number of gNBs in LOS conditions for the AGV. Table 9 shows the localization uncertainty models based only on the area in which the AGV is located.

Table 8: Localization uncertainty models for the InF-SH scenario of the port area. The models are reported based on the number of gNBs deployed and on the number of LOS gNBs

	Deployment configuration			
	1gNB	2 gNBs	3 gNBs	4 gNBs
0 LOS	$\boldsymbol{\mu} = [-6.148, 3.072]$ $\boldsymbol{\Sigma} = \begin{bmatrix} 351.035 & -213.484 \\ -213.484 & 237.427 \end{bmatrix}$ $n = 1787$	$\boldsymbol{\mu} = [3.367, -3.699]$ $\boldsymbol{\Sigma} = \begin{bmatrix} 97.681 & -14.168 \\ -14.168 & 25.938 \end{bmatrix}$ $n = 1378$	$\boldsymbol{\mu} = [1.651, -3.772]$ $\boldsymbol{\Sigma} = \begin{bmatrix} 84.418 & -18.679 \\ -18.679 & 23.279 \end{bmatrix}$ $n = 1378$	$\boldsymbol{\mu} = [6.101, -2.475]$ $\boldsymbol{\Sigma} = \begin{bmatrix} 49.869 & -13.391 \\ -13.391 & 16.170 \end{bmatrix}$ $n = 1075$
1 LOS	$\boldsymbol{\mu} = [-0.346, 0.279]$ $\boldsymbol{\Sigma} = \begin{bmatrix} 12.718 & -12.096 \\ -12.096 & 25.743 \end{bmatrix}$ $n = 2509$	$\boldsymbol{\mu} = [-1.193, 0.063]$ $\boldsymbol{\Sigma} = \begin{bmatrix} 59.207 & 16.826 \\ 16.826 & 10.240 \end{bmatrix}$ $n = 794$	$\boldsymbol{\mu} = [-2.631, 0.927]$ $\boldsymbol{\Sigma} = \begin{bmatrix} 15.329 & -2.564 \\ -2.564 & 5.887 \end{bmatrix}$ $n = 487$	$\boldsymbol{\mu} = [-3.454, -1.832]$ $\boldsymbol{\Sigma} = \begin{bmatrix} 15.088 & 12.156 \\ 12.156 & 15.545 \end{bmatrix}$ $n = 688$
2 LOS		$\boldsymbol{\mu} = [-0.127, -0.074]$ $\boldsymbol{\Sigma} = \begin{bmatrix} 0.863 & 0.161 \\ 0.161 & 0.563 \end{bmatrix}$ $n = 2124$	$\boldsymbol{\mu} = [-1.304, -0.400]$ $\boldsymbol{\Sigma} = \begin{bmatrix} 6.479 & 1.611 \\ 1.611 & 1.172 \end{bmatrix}$ $n = 307$	$\boldsymbol{\mu} = [0.995, 0.262]$ $\boldsymbol{\Sigma} = \begin{bmatrix} 2.660 & 0.895 \\ 0.895 & 0.528 \end{bmatrix}$ $n = 102$
3 LOS			$\boldsymbol{\mu} = [-0.092, -0.077]$ $\boldsymbol{\Sigma} = \begin{bmatrix} 0.485 & 0.025 \\ 0.025 & 0.275 \end{bmatrix}$ $n = 2124$	$\boldsymbol{\mu} = [-0.574, -0.260]$ $\boldsymbol{\Sigma} = \begin{bmatrix} 2.216 & 0.910 \\ 0.910 & 0.889 \end{bmatrix}$ $n = 307$
4 LOS				$\boldsymbol{\mu} = [-0.080, -0.060]$ $\boldsymbol{\Sigma} = \begin{bmatrix} 0.381 & 0.099 \\ 0.099 & 0.248 \end{bmatrix}$ $n = 2124$

$\boldsymbol{\mu}$: mean vector $\boldsymbol{\Sigma}$: covariance matrix n : number of trajectory samples available to fit the model

Table 9: Localization uncertainty models for the RMa scenario of the port area. The models are reported based on the number of gNBs deployed and on the number of LOS gNBs

	Deployment configuration			
	1 gNB	2 gNBs	3 gNBs	4 gNBs
0 LOS				
1 LOS	$\boldsymbol{\mu} = [-0.642, -1.990]$ $\boldsymbol{\Sigma} = \begin{bmatrix} 9.095 & 19.459 \\ 19.459 & 89.809 \end{bmatrix}$ $n = 5961$			
2 LOS		$\boldsymbol{\mu} = [-0.016, -0.035]$ $\boldsymbol{\Sigma} = \begin{bmatrix} 0.527 & 0.128 \\ 0.128 & 1.857 \end{bmatrix}$ $n = 5961$		
3 LOS			$\boldsymbol{\mu} = [0.007, -0.073]$ $\boldsymbol{\Sigma} = \begin{bmatrix} 0.989 & -1.080 \\ -1.080 & 3.569 \end{bmatrix}$ $n = 5961$	
4 LOS				$\boldsymbol{\mu} = [-0.003, -0.049]$ $\boldsymbol{\Sigma} = \begin{bmatrix} 0.290 & 0.099 \\ 0.099 & 0.871 \end{bmatrix}$ $n = 5961$

$\boldsymbol{\mu}$: mean vector $\boldsymbol{\Sigma}$: covariance matrix n : number of trajectory samples available to fit the model

Table 10: Localization uncertainty models for the RMa scenario and for the InF-SH scenarios of the port area for different deployments without considering the number of LOS gNBs

.

	Deployment configuration			
	1gNB	2 gNBs	3 gNBs	4 gNBs
InF Area	$\boldsymbol{\mu} = [-2.760, 1.441]$ $\boldsymbol{\Sigma} = \begin{bmatrix} 161.576 & -99.774 \\ -99.774 & 115.658 \end{bmatrix}$ $n = 4296$	$\boldsymbol{\mu} = [0.797, -1.212]$ $\boldsymbol{\Sigma} = \begin{bmatrix} 45.948 & -4.396 \\ -4.396 & 13.410 \end{bmatrix}$ $n = 4296$	$\boldsymbol{\mu} = [0.093, -1.172]$ $\boldsymbol{\Sigma} = \begin{bmatrix} 31.278 & -8.277 \\ -8.277 & 11.653 \end{bmatrix}$ $n = 4296$	$\boldsymbol{\mu} = [0.916, -0.955]$ $\boldsymbol{\Sigma} = \begin{bmatrix} 25.730 & -3.141 \\ -3.141 & 7.896 \end{bmatrix}$ $n = 4296$
RMa Area	$\boldsymbol{\mu} = [-0.635, -1.962]$ $\boldsymbol{\Sigma} = \begin{bmatrix} 9.092 & 19.319 \\ 19.319 & 89.646 \end{bmatrix}$ $n = 6000$	$\boldsymbol{\mu} = [-0.008, -0.034]$ $\boldsymbol{\Sigma} = \begin{bmatrix} 0.653 & 0.156 \\ 0.156 & 1.852 \end{bmatrix}$ $n = 6000$	$\boldsymbol{\mu} = [0.007, -0.074]$ $\boldsymbol{\Sigma} = \begin{bmatrix} 0.984 & -1.073 \\ -1.073 & 3.547 \end{bmatrix}$ $n = 6000$	$\boldsymbol{\mu} = [-0.000, -0.061]$ $\boldsymbol{\Sigma} = \begin{bmatrix} 0.507 & -0.221 \\ -0.221 & 1.328 \end{bmatrix}$ $n = 6000$

μ : mean vector Σ : covariance matrix n : number of trajectory samples available to fit the model

3.2.1.3 Results

The localization uncertainty models described above have been imported in the PoC system to simulate the behaviour of AGVs when the localization information input to the navigation system is affected by errors. Case with 4 gNB has been considered, as it shows enough accuracy to allow the AGVs to navigate without excessive deviations.

A set of 66 missions have been simulated and AGV paths has been plotted. Next figure shows an example of a mission (the path is covered in both directions) where deviations from ideal trajectory is self-evident

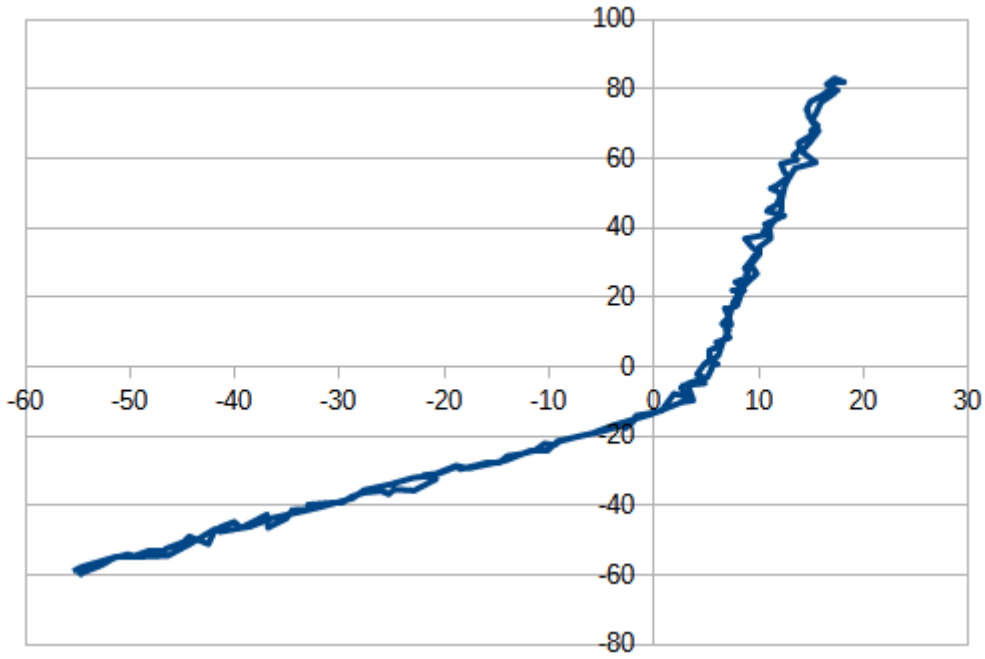


Figure 28: AGV path affected by positioning errors

The following figure instead shows the distribution of deviation of AGV position versus the ideal one along x and y axis.

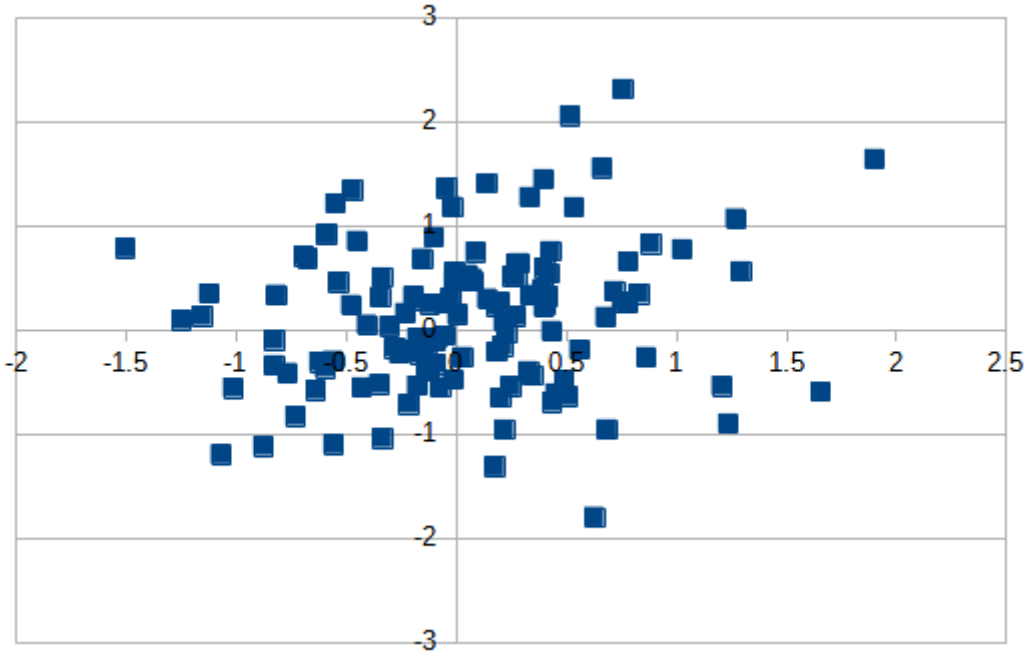


Figure 29: x and y distribution of errors w.r.t. ideal trajectory (values in meters)

3.2.1.3.1 Effect of latency

The previous results are not considering latency due to the calculation and exposure of localization data provided by the network. In fact, the localization information provided by the network is not presented instantaneously to the navigation system, but with a certain delay; the effect of this delay (that is not fixed, but with its own statistical distribution) can be seen as an additional error to be sum to the positioning error provided by the system.

An additional set of tests has been prepared in the VR simulation environment considering the effect of latency. The distribution of latency values has been taken from research performed in [28], In this work it is presented the design, implementation and evaluation of the 5G Location Management Function (LMF), the key network function in the 5G core for localization services. The implementation complies with the 3GPP standard (Release 16) and OpenAirInterface [29], the currently most advanced framework that implements a full 5G-New Radio stack.

The results are summarized in the following figure, where IPG are localization inter-packet gap, i.e., the time between two consecutive localization requests to the LMF (Localization Management Function).

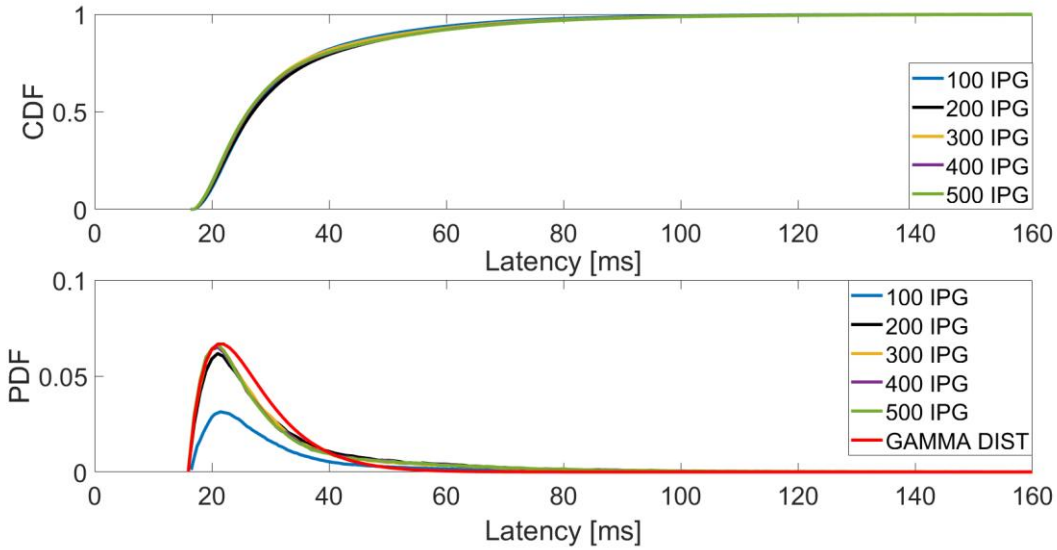


Figure 30: Distribution of latency and approximation by gamma distribution

According to [28], the latency distribution (with IPG >= 200 ms) can be approximated with a gamma distribution – it has been modelled this way in the simulator, that therefore is able to also reproduce this impairment in its AGV navigation emulation.

To be noted that, as the AGV maximum speed is typically 8 m/s, 100 ms of latency can cause a maximum additional error of 80 cm (see Figure 29 for a comparison with radio positioning error).

3.2.1.4 Updates on VR environment and navigation system

3.2.1.4.1 Specification on VR environment

The VR environment is based on the Unity3D professional game engine, and it is developed using the related editor. Models used in the simulated environment are taken from the Unity Asset Store and other 3D models libraries available on the web. The fundamental element is the GameObject that represents characters, properties, and scenery container for Components implementing the functionalities. Components characterize the GameObject making it a fixed part of the environment, (characters, lights, vehicles, etc.). Unity has many different built-in component types (e.g., 3D mesh model, rigid body, colliders), and it is possible to design own components using the Unity Scripting API. Programming is done in C#.

3.2.1.4.2 Freight sorting

The algorithm utilized for freight sorting is derived from the 3D bin packing algorithm, EB_AFIT, developed by the US Air Force.

Freights are grouped into categories based on cubic volume: Large, Medium and Small. The storage area is partitioned into areas with corridors in between to allow access to all pieces of freight. The algorithm considers constraints of having vehicles moving and operating in the corridors

Freights can be stacked depending on their characteristics (e.g., a heavy weight freight can't be placed over another one).

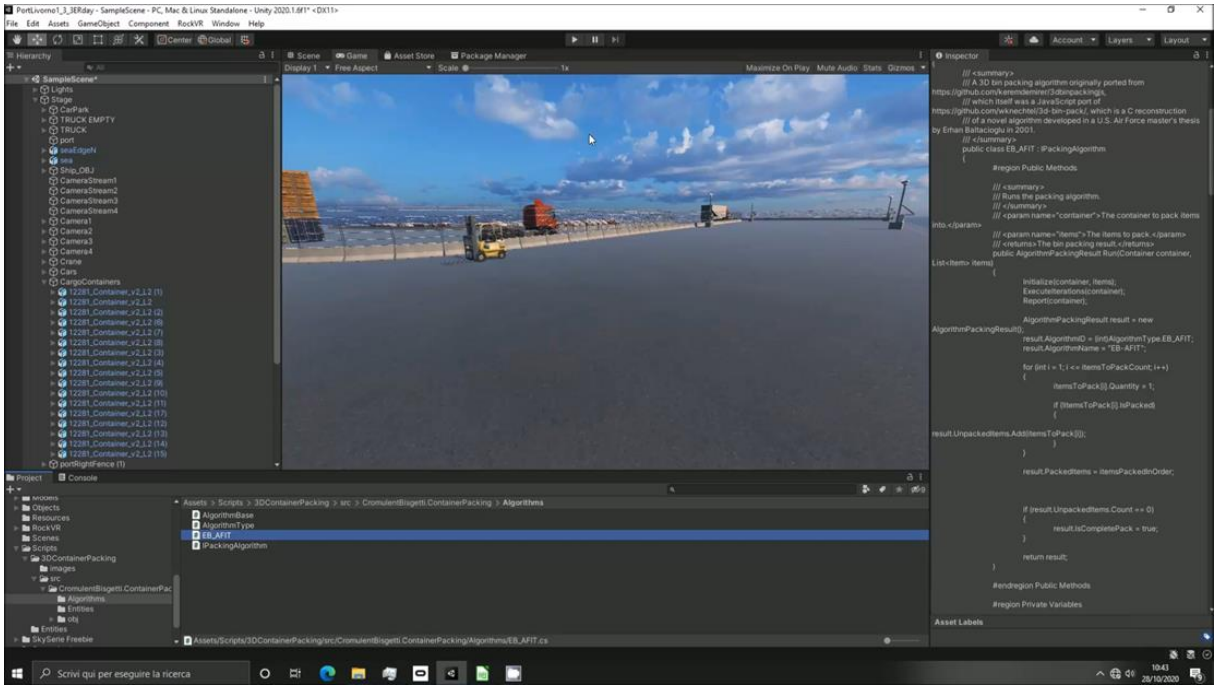


Figure 31: VR environment (Unity)

3.2.1.4.3 AGV control system

In the simulation the AGV movement can be controlled in two different ways to evaluate the performance of the guidance system:

- Using the LOCUS positioning
- Using LIDAR for collision avoidance

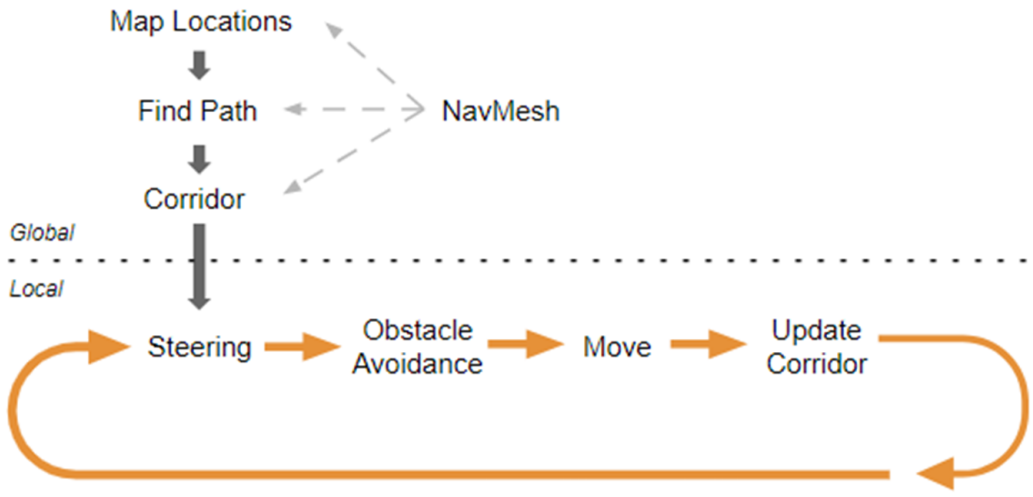


Figure 32: Navigation system flow

3.2.1.4.4 Navigation system

The navigation system defines walkable areas in the scene. The walkable areas define the places in the scene where the agent, described as a cylinder, can stand, and move. The walkable area is built automatically from the geometry in the scene by testing the locations where the agent can stand.

Locations are connected to a surface laying on top of the scene geometry, modelled as a “navigation mesh” (NavMesh [30]).

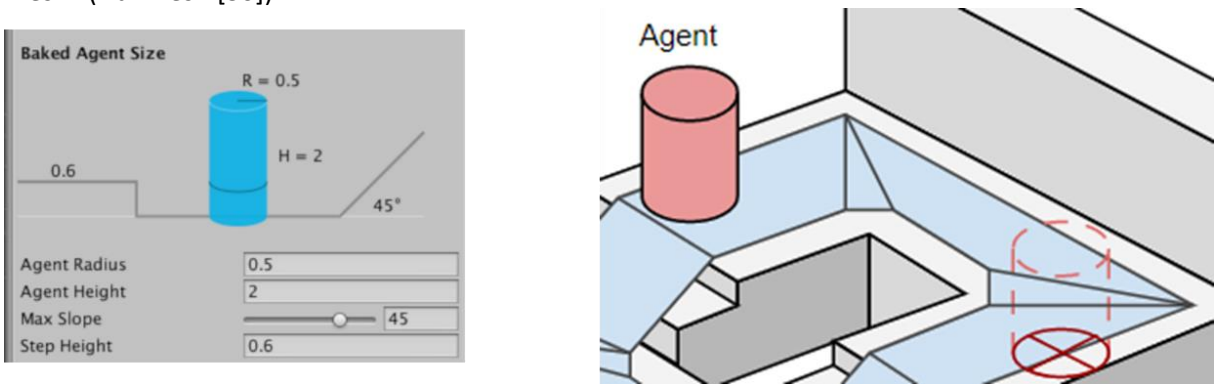


Figure 33: Agent and Navigation Mesh

Finding a path: given start and destination points, it maps the start and destination locations to their nearest polygons. A* algorithm is used to search a path from the start location until the destination polygon is reached. Then it calculates the shortest path on the NavMesh working on a graph of connected nodes. The sequence of polygons gives the moving path to follow. The agent will reach the destination by always steering towards the next visible corner of the corridor.

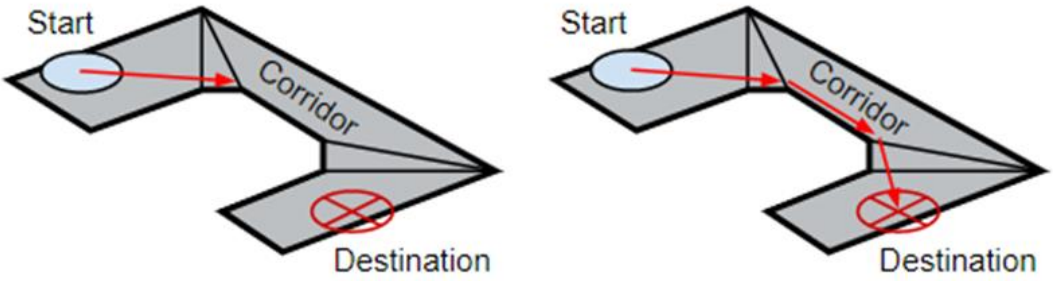


Figure 34: Agent finding path

3.2.1.4.5 Steering logic

The steering logic takes the position of the next corner and based on that figures out a desired direction and speed needed to reach the destination using the desired speed to move the agent can lead to collision with other agents.

Obstacle avoidance chooses a new velocity which balances between moving in the desired direction and preventing future collisions with other agents and edges of the navigation mesh. Reciprocal Velocity Obstacles (RVO) predict and prevents collisions.

3.3 Proof of Concept 3: People mobility and flow monitoring

3.3.1 Storyline 1*:* Flow Monitoring and Management in Large Venues

3.3.1.1 Concept

Urban areas embody mobility flow patterns that can be discovered with the use of unsupervised learning methods and be exploited for marketing or network configuration purposes. These patterns

can further be complemented with other relevant flow monitoring analytics the ones described in the following storylines (indicatively, path detection, trajectory prediction, etc.) and presented in different contexts.

3.3.1.2 End-to-end execution

In previous deliverables D5.1 [31], D5.2 [32], D5.3 [4] and D5.4 [2], there has been extensive analysis of the UCs involved in this scenario. In this document the end-to-end execution of the microservices will be further described. The pipeline consists of three distinct ML/analytics-based microservices as shown in Figure 35. More specifically, the "POI detection" function for the detection of Points of Interests (POIs), and the "Shape Correlation" function, which maps the UE IDs to the various shapes (in this case POIs). Lastly, the relevant flow monitoring profiling analytics in support of a smart venue/marketing context are calculated.

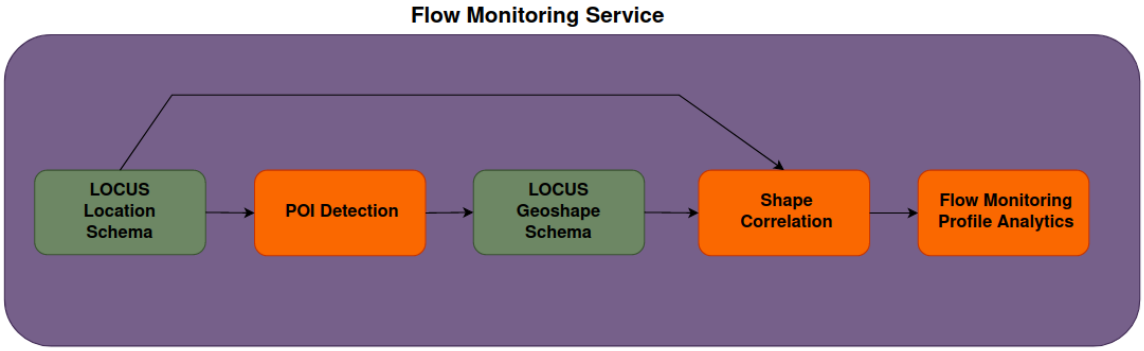


Figure 35: Flow monitoring analytics service pipeline for PoC#3 – 1st scenario

Each of the aforementioned microservices consists of three separate processing stages, namely the data pre-processing functions, the ML/Predictive modelling functions and post -processing/analytics acquisition functions. In the next paragraphs, the data flow of the whole pipeline is described.

POI Detection

The Point Detection service is responsible for identifying flow patterns between regions in a certain urban area. Since the service does not need any network related features (unless utilized for network management purposes), the input contains only the positioning information produced by a fingerprinting model persisted in the relevant module (Table 11).

Table 11: Input data of POI detection service

	timestamp	imei	lat	lon
0	24-08-20 13:30	1	37.98264687	23.73373422
1	24-08-20 13:30	2	37.98503855	23.73875045
2	24-08-20 12:30	3	37.98308098	23.73314582
3	24-08-20 12:30	4	37.98308098	23.73314582
4	24-08-20 12:30	5	37.98126553	23.73265727

The pre-processing here is minimal since the input table comes directly from the assumed fingerprinting service. There is only one test function to ensure that data types are not corrupted. The next phase includes the splitting of dataset on timestamps groups on an hourly basis and the execution of clustering algorithms of the UE coordinates for each time period. The frequency of the execution may be set to vary. The result of the function is the formation of spatial clusters regions that correspond to time evolving shapes and specifically POIs (POI shapes correspond to convex hulls of the UEs included in the POIs). After this execution an example of the data produced is presented in Table 12.

Table 12: Output data of POI detection service

	timestamp	lon	lat	shape_id
0	24-08-20 13:30	23.733734	37.982647	POI_0
1	24-08-20 13:30	23.731511	37.968621	POI_3
2	24-08-20 13:30	23.738750	37.985039	POI_0
3	24-08-20 12:30	23.733146	37.983081	POI_0
4	24-08-20 12:30	23.733146	37.983081	POI_0

Shape Correlation

Each coordinate pair is now marked with its POI label and related convex hull calculated from the set of UE positions of each POI that form its shape. This function maps the UE position at various times to specific POIs, allowing for calculation of new features KPIs that may relate to mobility aspects and could be used for vertical applications, such as marketing. In this storyline for each POI, two new features are calculated in the post processing phase. The first one is “Footfall per Minute” and the second is the “average stay duration in seconds”. Those features will be used to create the Marketing Profile of the POIs, with tags as Advertisement, Events, Socializing, Shopping or Other. Thus, the final output of the service is shown in:

Table 13: Indicative output of shape correlation function (JSON format)


```
"POIs": {
"kiosk_3_material=outdoor": {
"timeIdentified": 1659704212200,
"footfallPerMinute": 14.649387478795068,
"avgStayDurationSecond": 238,
"profile": {
"Advertisment": "Low",
"Events": "Low",
"Socializing": "Medium",
```

"Shopping": "Medium",

"Other": "Medium"

},

"poiName": "kiosk_3_material=outdoor",

"poiTags": ["LOCUS", "predictive"]

},

"highway_2_material=car": {

"timeIdentified": 1659704212200,

"footfallPerMinute": 12.657271175771186,

"avgStayDurationSecond": 55,

"profile": {

"Advertisment": "Medium",

"Events": "Medium",

"Socializing": "High",

"Shopping": "Low",

"Other": "Unknown"

} …

The flow of the POIs, as well as the Paths that are generated by Transportation Storyline UEs are stored as a joint output table of the two services that will represent all the correlations of two different geo-shapes along with the timestamp of the correlation.

Pedestrian Trajectory Prediction

Predicting the trajectory, or future position of a pedestrian in an urban environment requires modelling the complex social interactions and preferences people make when moving (walking side by side, avoiding collisions or narrow areas, etc), the pedestrian trajectory prediction model (described in [32] Section 2.1.1) performs such modelling using graph convolutional neural networks and spatial encodings. This service takes as input the location of pedestrians in a given location over several time steps. With this raw input the service generates a graph representing the scene for input into the model.

The raw input data required matches the output of the fingerprinting service (timestamp, pedestrian / device id, latitude, longitude).

The latest data is retrieved I/O module where it is aggregated into distinct frames (10 – 30 seconds long) and processed into a graph.

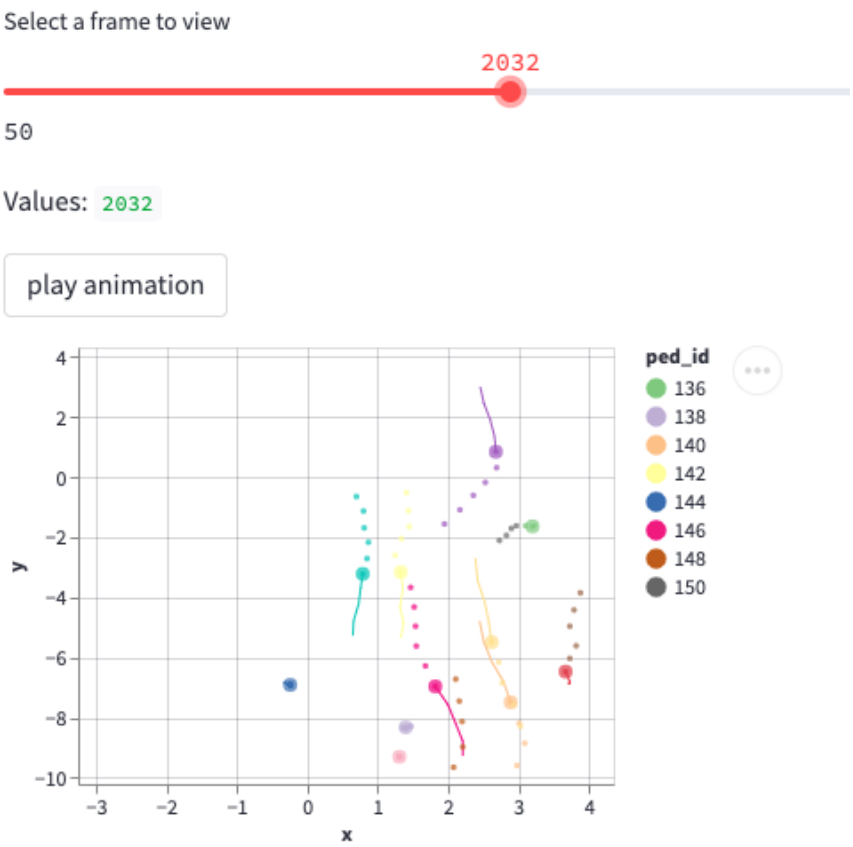


Figure 36: Simple UI showing the results of the pedestrian trajectory predictions – dotted lines show predicted trajectory of each pedestrian

The model can be retrained based on new data available in the I/O module before generating predictions. The model outputs include the future positions of all pedestrians present in the scene (with enough timesteps of data) along with data on the confidence intervals of such predictions.

3.3.1.3 Mapping on the platform

The functional blocks composing this service include:

- **The API Catalogue** to offer the analytics service capability to platform users, and trigger its automated activation in the platform.
- **The Analytics Coordinator** for managing the analytics service decomposition into individual virtualized analytics functions implemented as one or more NFV Network Services (to be then orchestrated by the LOCUS MANO).
- **The LOCUS Virtualization Management and Orchestration** (MANO) which is used for automatic configuration, orchestration, and operation of the different functions of the service.
- **The LOCUS SDK** and specifically the "Model I/O" module is used for communication with the LOCUS Persistence Module (data read/write).

- **The LOCUS Persistence Module** that persists both incoming data used by the various services, but also the results of the of those services.
- **The Data Operations Controller** (based on Apache Airflow [26]), which is responsible for orchestrating the pipeline.
- **The Service Discovery Module** (based on Consul [3]) that constitutes the service registry; and
- **The LOCUS API Gateway**, responsible for the exposure of this service to external 3rd parties.

A UI for the purposes of demonstration complements the aforementioned components, acting as a 3rd party application. For the case of trajectory prediction, the service is exposed at the API Gateway.

3.3.1.4 Results

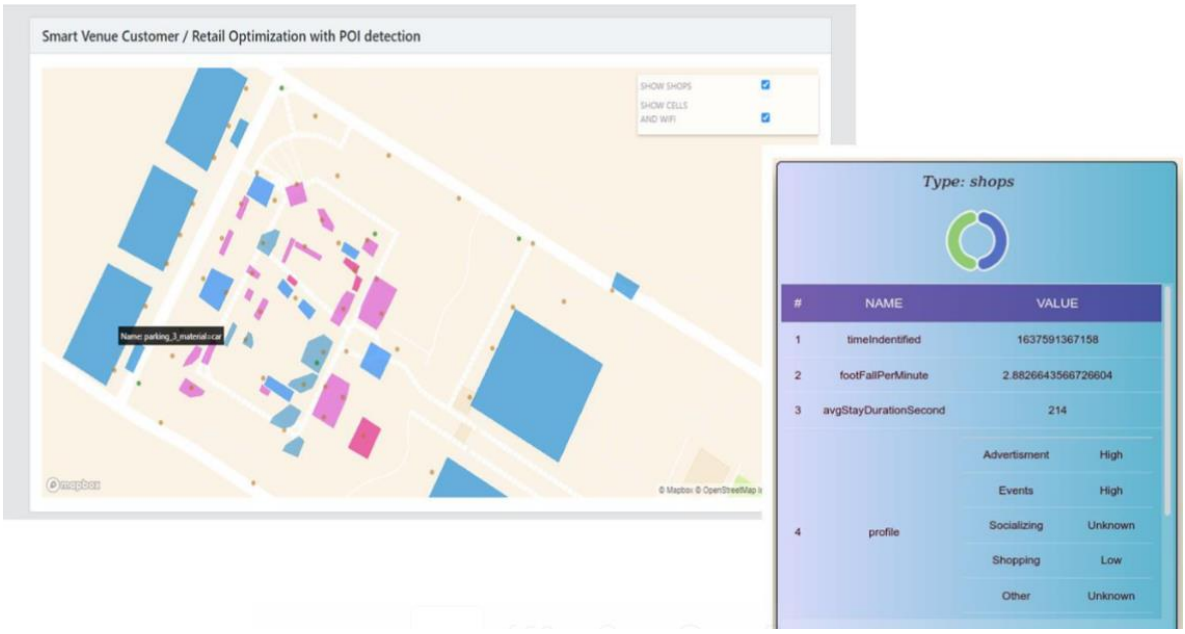


Figure 37: Example view of flow monitoring and management in venues

The experiments indicate that this pipeline can effectively discover spatial clusters in urban dense areas and their evolution in time for a specific period. The results offered by this microservice can be exploited for marketing and commercial purposes.

3.3.2 Storyline *2:* Crowd mobility analytics

3.3.2.1 Concept

This storyline extends people and crowd mobility analytics, such that additional crowd mobility analytics behaviours are considered based on the wireless fingerprints in the environment. In particular, this storyline considers extracting the crowd size and the group behaviours (e.g., people gatherings, social groups) in a crowd. It can complement the PoC with the group inference technology. The group inference would bring benefits to various domains such as smart environments where the characteristics of the gatherings would affect the services provided to the customers. For instance, understanding crowdedness and people groups at an airport would enable real-time adjustments such as opening new gates or transferring passengers efficiently in the environment. Furthermore, contact

tracing in controlled environments such as hospital can be provided with high resolution and long-term measurements.

3.3.2.2 End-to-end execution

In previous deliverables D5.1 [31], D5.2 [32], D5.3 [4] and D5.4 [2], there has been extensive descriptions of the crowd mobility analytics as a UC in the localization analytics including results with KPIs such as group inference accuracy using pairwise and Jaccard similarities. Out of functionalities of the use case, group inference (and inherently including crowd size estimation) is considered in the proof-of-concept due to its maturity and confidence-based experiments conducted through the controlled dataset with Bluetooth advertisement packages. Furthermore, the virtualization pipeline and service aspects have been highlighted in the previous deliverables.

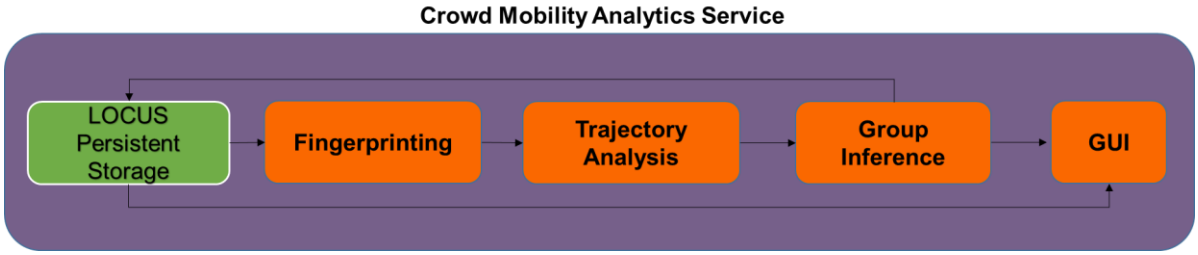


Figure 38: Crowd mobility analytics service pipeline for PoC#3 – 2nd scenario

Figure 38 shows the crowd mobility analytics service at a high level, where the initial stage includes the wireless fingerprinting. The second stage includes the wireless fingerprint trajectories, and the third stage includes the analytics for the people group inference.

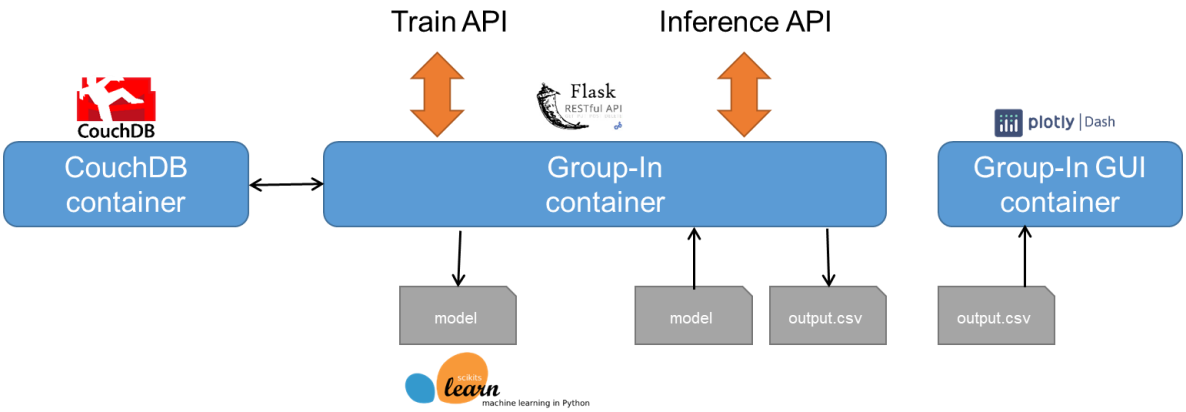


Figure 39: Initial adoption of the Group-In prototype for containerization.

The prototype system for group inference, namely Group-In, is adopted to the LOCUS platform to integrate with the PoC (see Figure 39). As a first step of adoption, the prototype is exposed with two new interfaces in the API: 1) Train API and 2) Inference API. These interfaces are run in the main "Group-In" container as a Flask service. Furthermore, the initial Group-In prototype is divided into three containers, the initial one for the container that includes a database (i.e. , CouchDB container),

the second one that includes the core fingerprinting and analytics functionalities (i.e., Group-In container). And the third one that includes the graphical user interface (i.e., Group-In GUI container). The main container includes the scikit-learn machine learning models and outputs a CSV file which can be consumed by the GUI container.

3.3.2.3 Mapping on the Platform

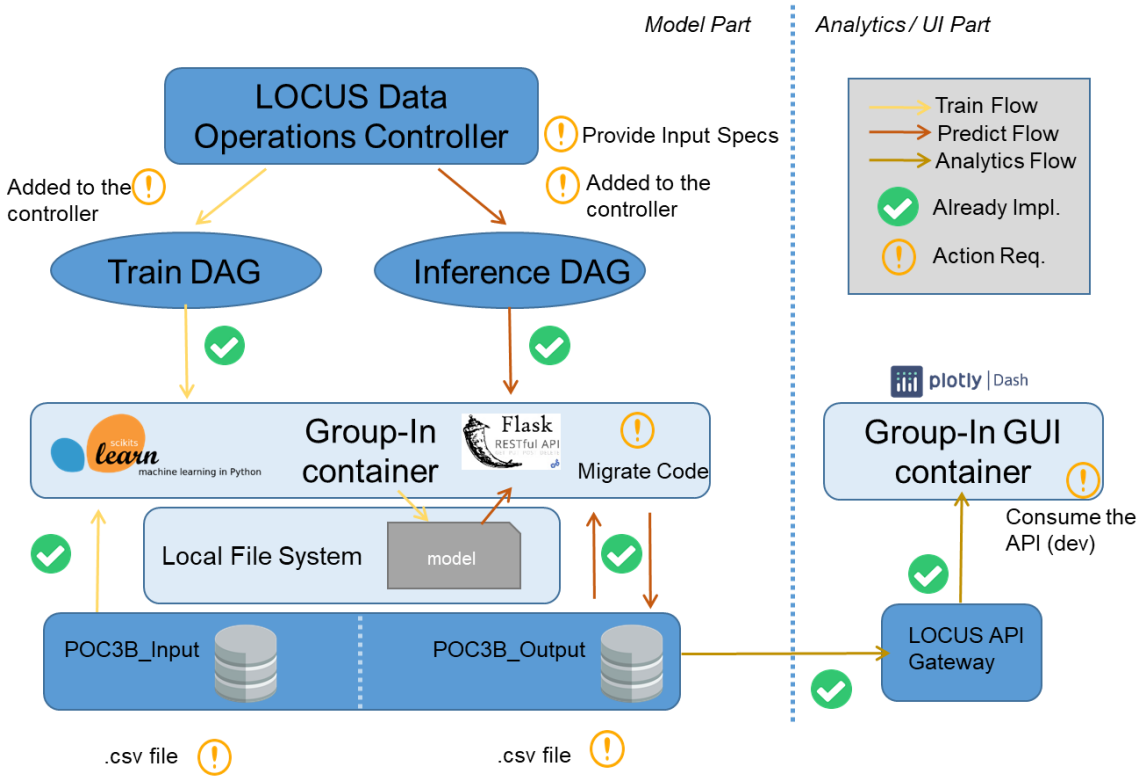


Figure 40: Mapping of the containers to the PoC platform.

Figure 40 illustrates the architecture for the actual integration of the analytics and the user interface. The architecture is created by NEC and Incelligent, considering the LOCUS operation and API functions of the Group-In container and the GUI container. The data is inputted to and queried from the persistent storage whereas the trained machine learning model is held on the local file system. The input specifications, data files, and the API requests are provided in the integration.

The data files include anonymous device data from the experiments where Bluetooth Low Energy beacons are carried in the controlled manners. Each experiment has also ground-truth values corresponding to the groups that the device belongs to, whereas the devices can be static or dynamic (e.g., moving groups). The machine learning models are configured using a small ground-truth data whereas the analysis is tested on various controlled experiments as described in [33].

3.3.2.4 Results

The experiments indicate that the Group-In pipeline can effectively identify people groups given the wireless fingerprints collected from the environment. The data analytics results include the size of the crowd in the vicinity (I.e., number of wireless devices detected), the number of groups and the size of

each inferred group. The GUI illustrating the crowd mobility analytics is included in the previous deliverables.

Figure 41. Group inference results from a mobile scenario: Straight-walking groups scenario. The results include two metrics for Jaccard and pairwise accuracy (from [33]).
Figure 42. Group inference results from another mobile scenario: Group imitating random walks. The results include two metrics for Jaccard and pairwise accuracy (from [33]).

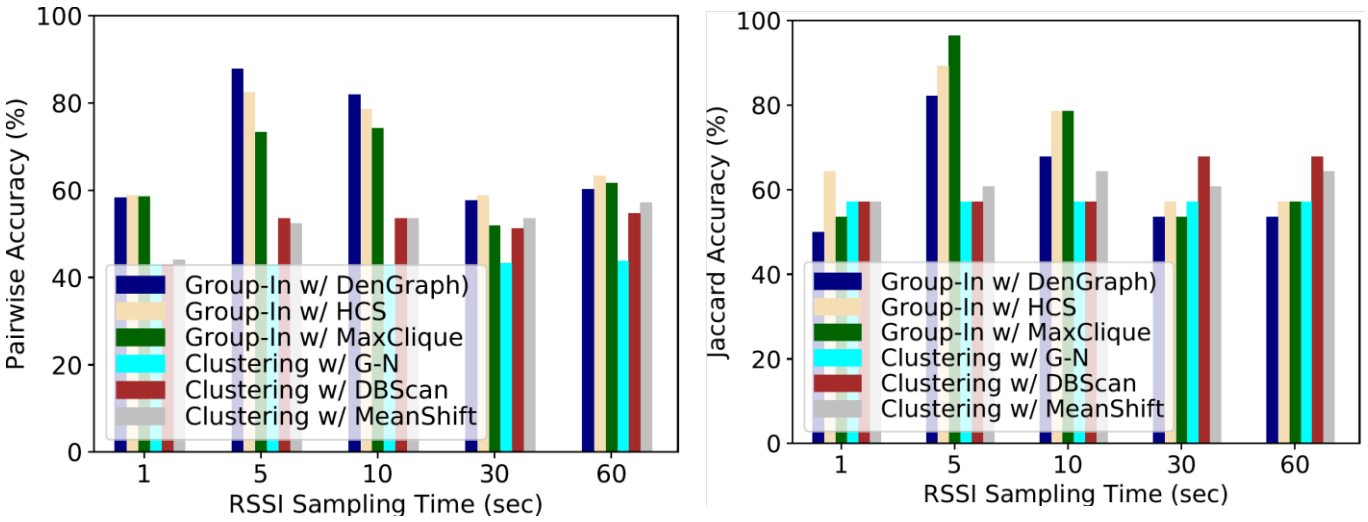


Figure 41: Group Inference Results, straight-walking

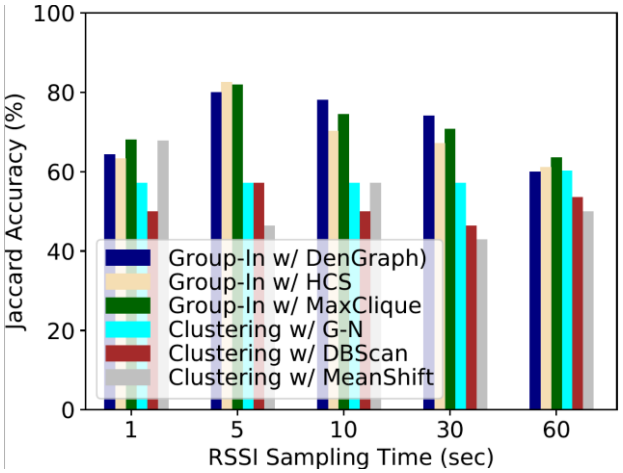


Figure 42: Group inference results, random walks

In the previous deliverables, Group-In results for given scenarios mainly for the static group detection. Furthermore, the results included (also in [33]) include various scenarios such as inter-group distances. The above figures include the group inference accuracy for two mobile scenarios: 1) Straight-walk and 2) Random-walk scenarios. For simplicity, two groups of devices are carried where the Group-In system aims to infer the number of groups and the devices that belong to the groups. The accuracies include Jaccard accuracy which is based on the actual sets of devices compared to the sets of devices that are inferred by the group inference. Moreover, pairwise accuracy considers every pair of devices and analyses if two devices belong to the same group or different group. Pairwise similarity score is calculated based on all pairwise comparisons. In the controlled experiments, the accuracies of group

inference reach to >95% for the straight-walk scenario given 5 second sampling time, whereas with longer sampling durations the accuracy suffers as the mobility behaviours are not captured in high resolution. For the random-walk scenario, the results reach to >85% for the same sampling time of 5 seconds.

3.3.3 Storyline 3: Transportation optimization

3.3.3.1 Concept

This storyline is closely related to Storyline 1, in the sense that it is based on the concept that urban areas formulate mobility patterns that can be discovered with the use of unsupervised learning methods and be exploited for other purposes. It can complement the POI Detection function described there and focuses on the detection of paths/trajectories. While both functionalities can be used for various purposes, Storyline 1 is placed in the context of a venue/market area UC, while this storyline refers to a transportation optimization context. More specifically, detecting patterns in terms of common trajectories/paths and identifying the mobility profiles (e.g. pedestrian or vehicular) can be further utilized to optimize public transportation, monitor traffic patterns by authorities and so on.

3.3.3.2 End-to-end execution

In previous deliverables D5.1 [31], D5.2 [32], D5.3 [4] and D5.4 [2], there has been extensive analysis of the UCs involved in this scenario. As in Storyline 1, it utilizes different microservices. The function pipeline consists of two distinct microservices, as presented in Figure 43. It implies that the original input is the positioning results persisted and coming from a localization enabler, namely "fingerprinting".

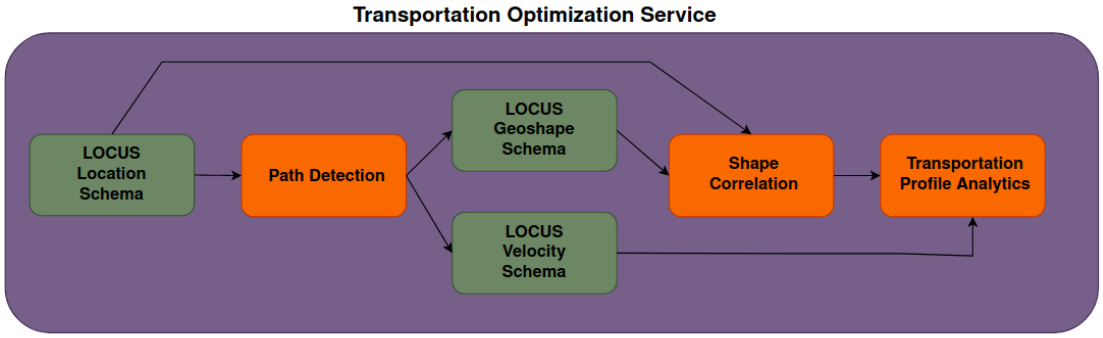


Figure 43: Transportation optimization analytics service pipeline for PoC#3 – 3rd scenario (Path detection)

All the microservices include a pre-processing stage, an ML/predictive stage and post-processing/analytics acquisition stage. The data flow of the whole pipeline described is presented in detail in the following subsections.

Path Detection

The Path Detection analytics service is responsible for identifying common mobility patterns in an outdoor urban area. As in the POI Detection service, Path Detection relies on spatial features, thus the input table contains only the geolocation information produced by the fingerprinting model as well as the relevant UE IDs and timestamps (see Table 14).

Table 14: Input data for path detection service

	timestamp	UE	lat	lon
0	24-08-20 13:30	1	37.98264687	23.73373422
1	24-08-20 13:30	2	37.98503855	23.73875045
2	24-08-20 12:30	3	37.98308098	23.73314582
3	24-08-20 12:30	4	37.98308098	23.73314582
4	24-08-20 12:30	5	37.98126553	23.73265727

Here the pre-processing involves again the casting to correct data types and the drop of null data values. Next, the individual trajectories of UEs are extracted from the dataset and some new features are created. More specifically, for each step of the trajectory a velocity and an azimuth shift are calculated. These, along with the coordinates pair of measurement are going to be fed to the clustering algorithm of the service. In this way, the spatial coordinates are augmented with mobility -related features that can reveal patterns about the movement of the mobiles. Table 15 shows example output.

Table 15: Path detection intermediate table

	UE	timestamp	lon	lat	azimuth	velocity
0	1	24-08-20 13:30	23.733734	37.982647	1.123	0.87
1	2	24-08-20 13:30	23.731511	37.968621	0.71	1.14
2	3	24-08-20 13:30	23.738750	37.985039	0.159	3.46
3	4	24-08-20 12:30	23.733146	37.983081	0.672	0.44
4	5	24-08-20 12:30	23.733146	37.983081	1.015	1.67

Then, timestamps are grouped on a preset basis (e.g. hourly) and the execution of clustering algorithms for each time period ensues. This results in the formation of spatial clusters regions that correspond to time evolving paths. After this execution, the data are stored as in Table 16, where paths a LineString objects:

Table 16: Output data of path detection service

	UE	timestamp	Shape_id	Geometry
0	1	24-08-20 13:30	Path_1	{"type": "LineString", "coordinates": [7.73, 10.37] …}
1	2	24-08-20 13:30	Path_1	{"type": "LineString", "coordinates": [4.42, 8.92] …}
2	3	24-08-20 13:30	Path_2	{"type": "LineString", "coordinates": [16.91, 7.84] …}

3	4	24-08-20 12:30	Path_3	{"type": "LineString", "coordinates": [8.354, 8.231] …}
4	5	24-08-20 12:30	Path_3	{"type": "LineString", "coordinates": [11.221, 9.12] …}

Shape Correlation

Each UE now is paired with one or more Linestring objects that corresponds to a common area Path for each timestep, along with the mobility features (velocity and azimuth). These features are exploited in the post-processing phase of the service to construct the mobility class of the UE (Table 17).

Table 17: Output data of shape correlation function for paths

Time-stamp	UE	Lat	Lng	RelativeX	RelativeY	Azimuth	Velocity_KMh	TransportationState	PathShape
24-08-20 13:30	1	37.9873	23.909502	21.0	72.0	nan	nan	Immobile	{"type": "LineString", "coordinates": [7.73, 10.37] …}
24-08-20 13:30	2	37.9872	23.9094	20.53	85.96	1.624562	0.399541	Pedestrian	{"type": "LineString", "coordinates": [4.42, 8.92] …}
24-08-20 12:30	3	37.9873	23.909502	21.0	72.0	nan	nan	Immobile	{"type": "LineString", "coordinates": [16.91, 7.84] …}
24-08-20 12:30	4	37.9873	23.909502	21.0	72.0	nan	nan	Immobile	{"type": "LineString", "coordinates": [8.354, 8.231] …}
24-08-20 12:30	5	37.9853	23.910	169.0	243.0	0.12	2,74	Vehicle	{"type": "LineString", "coordinates": [11.221, 9.12] …}

The Paths that are generated by Transportation Optimization Service, as well as the POIs discovered by the flow monitoring service are stored as a joint output table of the two services that will represent all the correlations of two different geo-shapes along with the timestamp of the correlation (see Table 18).

Table 18: Joint output table for POI & Path geo-shapes

	timestamp	UE	POI id	PATH_id
0	24-08-20 13:30	1	POI_112	None
1	24-08-20 13:30	2	POI_31	PATH_23
2	24-08-20 12:30	3	None	PATH_12
3	24-08-20 12:30	4	POI_42	PATH_11
4	24-08-20 12:30	5	POI_42	None

3.3.3.3 Mapping on the platform

As in Storyline 1, the platform components utilized for the deployment of this storyline include the API Catalogue, the Analytics Coordinator, the LOCUS Virtualization Management and Orchestration (MANO), the LOCUS SDK and specifically the "Model I/O" module, the LOCUS Persistence Module, the Data Operations Controller, the Service Discovery Module and the LOCUS API Gateway.

A UI for the purposes of demonstration complements the aforementioned components, acting as a 3rd party application.

3.3.3.4 Results

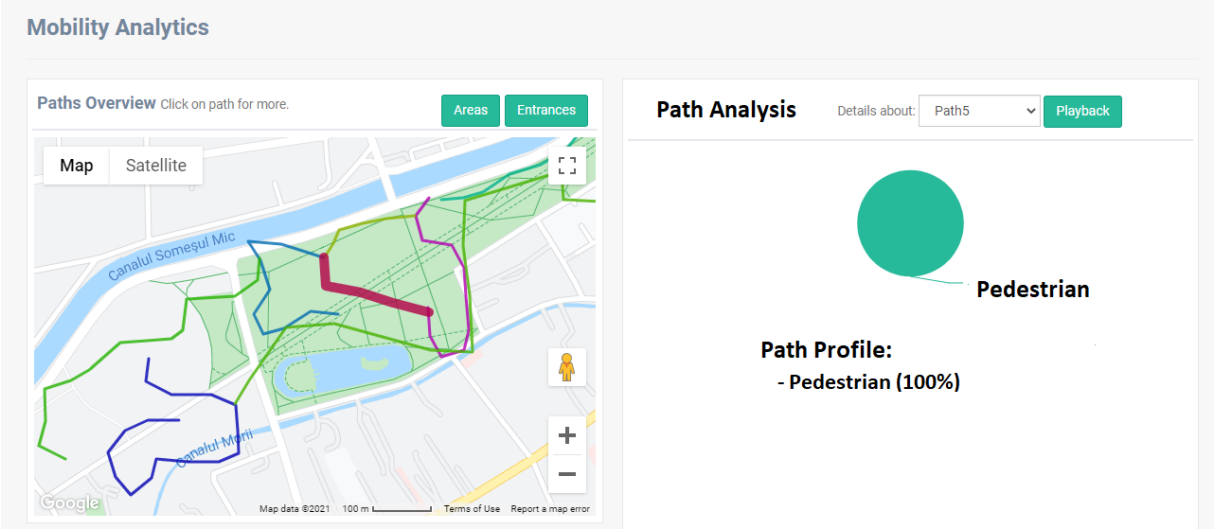


Figure 44: Indicative view for transportation optimization analytics

The experiments indicate that this pipeline can effectively discover useful common paths urban dense areas for a specific time period. The results offered by this microservice can be exploited for transportation optimization and other purposes.

3.3.4 Storyline 4: Collision Detection for Self-driving Car

3.3.4.1 Concept

The concept of this storyline is Artificial-Intelligence (AI) techniques, more particularly supervised learning can provide real-time, short-term trajectory predictions than can enable a collision detection service to automatically warn about possible collisions in indoor or outdoor areas. Since no such collision data are available, the scenario focuses mostly on the applicability of the analytics pipeline used, the deployment/platform aspects and integration of such services at the API Gateway level. No specific UI is therefore employed. The models are trained and implemented using UMA testbed data where people move around a designated area and need to avoid collisions.

3.3.4.2 End-to-end execution

The collision detection was presented at a theoretical level in WP5, however the functionality for trajectory prediction has been extensively analysed with various algorithms approaches and very good results. Here, we focus on one specific algorithm presented in WP5 deliverables, however -due to the self-contained nature of the microservices- any such function could be part of the service pipeline of this storyline. In this document the end-to-end execution of the micro-service will be further described. Three analytics microservices composed this storyline as shown in the in the following figure (Figure 45).

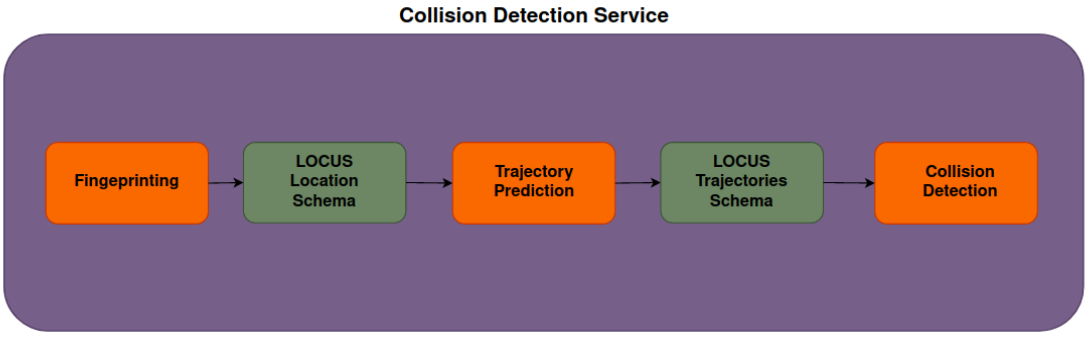


Figure 45: Collision Detection analytics service pipeline for PoC#3 – 4th scenario

Fingerprinting

Table 19 represents the input data table for the first service of the pipeline, the Fingerprinting model. This is a supervised neural network that predicts the location of mobile given the input (Wi-Fi/5G) measurements (e.g. in this cases RSSI, RSRP) from 1 to N access points/cells. The first two columns correspond to the timestamp of the measurement and the ID of the mobile UE. Its goal is to reproduce the last two columns referring to latitude and longitude (i.e. the position of the UE) given the input described.

Table 19: Input data for fingerprinting service

	timestamp	imei	RSSI_1	…	RSRP_N	lat	lon
0	24-08-20 13:30	1	-91.226	...	-90.334	37.98264687	23.73373422
1	24-08-20 13:30	2	-92.542	...	-86.283	37.98503855	23.73875045

2	24-08-20 12:30	3	-92.148	...	-86.557	37.98308098	23.73314582
3	24-08-20 12:30	4	-93.828	...	-92.670	37.98308098	23.73314582
4	24-08-20 12:30	5	-80.927	...	-81.891	37.98126553	23.73265727

Starting with the pre-processing steps, the custom I/O module (part of LOCUS SDK) is invoked to retrieve recent data from the database. Next steps include data cleaning, casting to appropriate data types, data split to train/ test/ validation subsets and data scaling to help the training procedure. The processing stage refers to training or inference, while the post-processing includes the inverse scaling of the predicted locations and the writing of the result to the database, again with the help of our custom I/O module.

Trajectory Prediction

The output of the Fingerprinting service -described in detail in Storyline 1- is fed to the Trajectory Prediction microservice. The pre-processing stage of this microservice includes the extraction of the trajectories from the UE predicted locations. Specifically, the data points are grouped based on the UE ID and then sorted in chronological order to form time series. Sequences with less than 20 data points are dropped, since the predictive phase will include the training of a sequential neural network, and the sequences must all have the same shape.

In the processing phase, 12 of the sequence data points of each produced sequence are kept for training the model while the rest will form the testing dataset. The model trained is a custom neural architecture inspired by the Transformer network described in D5.2 [32]. The attention blocks of the Transformer network are combined with GRU recurrent units that reduce significantly the hyperparameters complexity of the architecture, as well as the latency of the training and inference procedures of the microservice.

The predicted trajectories are then unravelled in coordinate points with a new projected timestamp produced by linear interpolation in the time dimension. Finally, they are transformed into LineString objects, for easier visualization and manipulation. As such, they are stored back to the LOCUS Persistence Module, as presented in Table 20 .

Table 20: Output data for trajectory prediction service

	User id	**Trajectory id**	**Starting time**	**Ending time**	**Geometry**
0	1	LineString 1	2021-10-21T12:14:23	2021-10-21T12:14:35	{“type”: “LineString”, “coordinates”: [7.861, 9.628] ...
1	1	LineString 2	2021-10-21T12:14:39	2021-10-21T12:14:50	{“type”: “LineString”, “coordinates”: [[12.109, 7.170]

2	2	LineString 3	2021-10-21T12:14:52	2021-10-21T12:15:03	{“type”: “LineString”, “coordinates”: [[9.05, 5.951]
3	3	LineString 4	2021-10-21T12:15:04	2021-10-21T12:15:15	{“type”: “LineString”, “coordinates”: [[14.82, 7.799]

Collision Detection

The final part of the service is the actual detection of possible collisions. Each produced trajectory, stored as LineString object, are marked with a starting and finishing timestamp. In that way, the processing phase can be executed in distinct groups of time which ensures that the memory efficiency of the service.

For every trajectory pair in the same group, a geometric intersection is calculated. If the intersection is not empty, the trajectory pair is labelled as a possible collision. Furthermore, if the time periods of the trajectories overlap, a collision time period is also calculated, as the maximum period from the starting and ending timestamps of the two trajectories. The maximum period is preferred as a safety valve in a worst-case scenario to ensure that a collision will be detected even if the approximation of the timestamps of trajectories lack accuracy.

The output of the service is shown in Table 21.

Table 21: Output data for collision detection service

	Collision Point	**Collision Start**	**Collision End**	**Colliding UE 1**	**Colliding UE 2**	**Colliding Trajectory 1**	**Colliding Trajectory 2**
0	{"type": "Point", "coordinates": [12.053, 6.951]}	2021-10-21T12:14:23	2021-10-21T12:14:47	3	2	1	2
1	{"type": "Point", "coordinates": [9.412, 8.12]}	2021-10-21T12:40:11	2021-10-21T12:40:18	3	4	1	3
2	{"type": "Point", "coordinates": [11.823, 8.47]}	2021-10-21T13:19:06	2021-10-21T13:19:21	1	8	2	3

3.3.4.3 Mapping on the platform

As in Storyline 1, the platform components utilized for the deployment of this storyline include the API Catalogue, the Analytics Coordinator, the LOCUS Virtualization Management and Orchestration (MANO), the LOCUS SDK and specifically the "Model I/O" module, the LOCUS Persistence Module, the Data Operations Controller, the Service Discovery Module and the LOCUS API Gateway. Lastly, the service will be exposed though the relevant API, however no UI will be delivered.

3.3.5 Storyline 5: Predicting Pedestrian trajectories in crowded indoor & urban environments

3.3.5.1 Concept

Due to the unique, social and collaborative way that humans move in a crowded environment, walking side by side, avoiding pinch points and negotiating with those crossing paths, it can be difficult to predict their future paths. Predicting where pedestrians will be in the near future can be of value to autonomous vehicles attempting to plan a route in a busy indoor environment, such as an autonomous mobility aid navigating an airport terminal. The goal of the autonomous vehicle may be to avoid getting in the way of pedestrians, and for this it is important to be able to predict where a pedestrian might go in the next steps.

Having short to medium term predictions of the trajectories of pedestrians could also be used for short term resource allocation in a shopping or security domain, providing fore warning of potential bottlenecks at checkouts or security checks.

3.3.5.2 End-to-end execution

Pedestrian Trajectory Prediction

Predicting the trajectory, or future position of a pedestrian in an urban environment requires modelling the complex social interactions and preferences people make when moving (walking side by side, avoiding collisions or narrow areas, etc.), the pedestrian trajectory prediction model (described in D5.2 [32] Section 2.1.1) performs, such modelling using graph convolutional neural networks and spatial encodings. This service takes as input the location of pedestrians in a given location over several time steps (8 by default). With this raw input the service generates a graph representing the scene for input into the model.

The raw input data required matches the output of the fingerprinting service (timestamp, pedestrian / device ID, latitude, longitude).

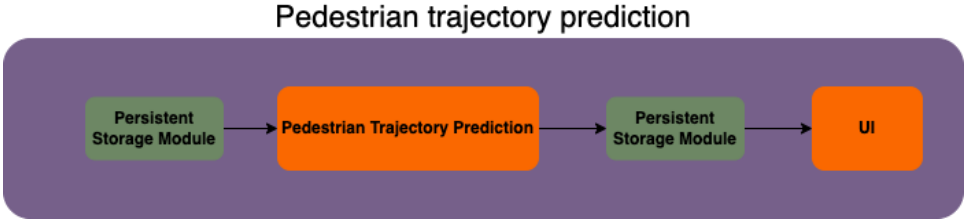


Figure 46: Pedestrian trajectory prediction service pipeline for PoC#3 – 5th scenario

The latest data is retrieved from the I/O module where it is aggregated into distinct frames (10 – 30 seconds long) and processed into a graph.

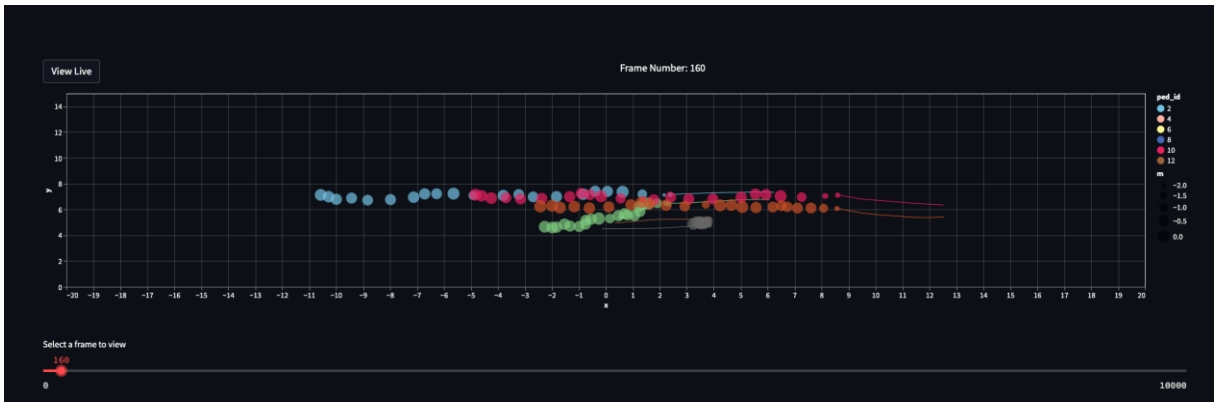


Figure 47: Simple UI showing the results of the pedestrian trajectory predictions – dotted lines show predicted trajectory of each pedestrian, with the size of the dot indicating the uncertainty in prediction.

The model can be retrained based on new data available in the I/O module before generating predictions. The model outputs include the future positions of all pedestrians present in the scene (with enough timesteps of data) along with data on the confidence intervals of such predictions.

Table 22: Pedestrian trajectory predictions output

Timestep	Pedestrian ID	Mean predicted Latitude	Mean predicted Longitude	Latitude confidence interval	Longitude Confidence interval
0	1	25.3734	47.2647	3.34	1.47
0	2	25.1511	47.8621	2.11	2.21
0	3	25.8750	47.5039	3.50	1.39
1	1	25.3146	47.3081	3.46	3.81
1	2	25.3146	47.3081	2.46	4.81
...	...	...	...	...	...

3.3.5.3 Mapping on the platform

This storyline makes use of the following platform components:

- The API Catalogue.
- The Analytics Coordinator.
- LOCUS Virtualization Management and Orchestration (MANO).

- LOCUS SDK with the Model I/O module.
- LOCUS Persistence Module.
- Service Discovery Module.
- LOCUS API Gateway.

A simple UI was developed for demonstration purposes which is deployed as a 3rd party application on the platform.

3.3.5.4 Results

The pedestrian trajectory prediction service produces accurate results in a range of scenarios, performing favourably compared to current state of the art techniques, in particular when dealing with social walking (walking side by side), for more details on the accuracy of the service see [32] Section 2.1.1.

4 Additional Demonstrations

Building on the experimental results obtained in WP3, three additional demonstrations have been conducted, which focus on the project's scientific advancements in terms of localization enablers, including beyond 5G, and thus are separated from the Proof-of-Concepts.

Specifically, two device-free localization demos leverage the technical achievements described in D3.8 [20]. A first demo shows the joint sensing and communication system developed in LOCUS for activity recognition and person identification using mm Wave experimentation platform (see Section 5.1.3 of D3.8 [20]). A second demo shows the multi-target tracking system developed in LOCUS and tested indoor with FCMW MIMO radars operating in the frequency range 76-78 GHz (see Section 5.1.6 of D3.8 [20]).

An additional demo shows the 5G Testbed based on Open Air Interface, developed in LOCUS and described in detail in the next subsection.

Note that three videos for the aforementioned demos have been prepared and recorded and will be available on the LOCUS website [34].

4.1 5G Positioning with SDR-based Open-source Platform

Many open-source platforms (such as OpenAirInterface, srsRAN, Aether) have gained popularity in the academic and industrial environment to implement and simulate the 3GPP stack. Such software platforms are able to run on general-purpose computing platforms and interfaces with a wide variety of software-defined radios (SDRs). Therefore, they represent an invaluable tool for research and development, enabling researchers to rapidly prototype new algorithms and paradigms and test them in real-world over-the-air conditions. Despite the long-term goal of enabling widespread use of these tools, to date their use still requires significant effort in terms of hardware and software set-up and configuration.

More specifically in the context of 5G positioning, some of these open-source platforms already implemented the reference signals dedicated to 5G positioning according to the latest 3GPP standards. However, their use is not immediate, in addition, the emulation of an entire positioning system would require setting up a network of multiple gNBs, which is costly and not easily available.

As a result, several research groups are forced to implement their systems with one or two radios and with long development times.

This subsection presents a simple but effective approach to using open-source platforms for 5G positioning, demonstrating both the main technical procedures and available resources, as well as how to overcome the limited availability of devices while achieving a localization performance assessment via simulation. We want to emphasize that our goal is not to provide a performance assessment of 5G localization, as the first real deployments are expected to be available soon, but to provide a simple, low-cost and effective tool that allows researchers and industry to experiment and validate algorithms and research results using measurements of real over-the-air transmitted 5G positioning signals, by taking into account standardized localization procedures and real-world operating conditions.

4.1.1 5G open-source platforms

We now describe the main existing open-source software platforms/tools for the implementation of a 5G Stand Alone (SA) mobile network infrastructure including RAN and Core network. The main features will be analysed focusing on the positioning aspects.

OpenAirInterface: OpenAirInterface (OAI) [29] provides implementations of 5G Core, gNB and UE compliant with NR Release 15 (with an additional subset of NR Release 16 features). The OAI source code is written in C to ensure real time performance and is distributed under the OAI Public License. Both gNB and UE implementations are compatible with Intel and ARM architectures running Linux distributions and support general purpose SDR platforms such as EXMIMO, BladeRF, USRP and LimeSDR. For real-time performance, several kernel and BIOS modifications are recommended, including installing a low-latency kernel and disabling power management and CPU frequency scaling features. In the last period, OpenAirInterface implemented both DL PRS and UL SRS positioning signals via dedicated development branches. This makes OAI the ideal candidate platform for testing 5G positioning.

srsRAN: Similar to OAI, srsRAN [35] provides software implementations of gNB and UE with features up to NR Release 15. Unlike OAI, it does not provide an implementation of 5GCN but uses Open5GS, an open-source 3GPP compliant 5G Core implementation written in C language. srsRAN is written in C and C++ and is distributed under the GNU AGPLv3 licence. srsRAN is compatible with the most popular Linux distributions (i.e. Ubuntu and Fedora) and like OAI is compatible with the most popular SDR platforms. It is a relatively more stable project compared to OAI but offers fewer functionalities, among which support for 5G positioning signals is missing.

Aether: Aether [36]is a project promoted by the Open Networking Foundation (ONF) for the simplified implementation of private cellular networks. Unlike OAI and srsRAN, Aether not only offers an implementation of the RAN and 5G Core, integrating SD-RAN [37] and SD-CORE [38] respectively, but also provides a control and orchestration interface for the RAN, a cloud edge platform with support for cloud computing, and a central cloud. The source code is released under the Apache 2.0 open-source license [39]. SD-RAN's main focus is to provide an O-RAN compliant micro-ONOS [40] based near-RealTime RAN Intelligent Controller (RIC) and a platform for xAPPS. SD-RAN relies on OAI gNB and UE implementations. Since we were not interested in Aether's additional features, we decided in our experimental setup to use OAI directly.

UERANSIM: is another well-known open-source project that enables very simple and functional implementation of UE and gNB. Unlike the previously mentioned platforms, UERANSIM is not SDR compatible and simulates the 5G NR radio interface between UEs and gNBs over the UDP protocol.

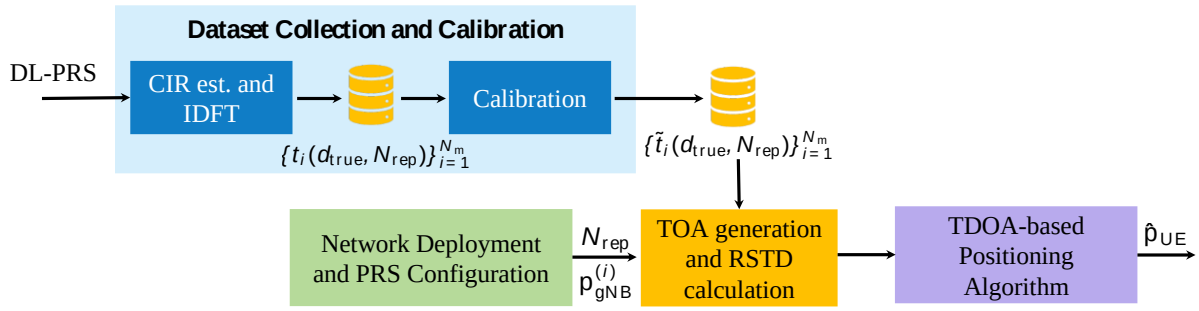


Figure 48: Illustration of the main step for dataset collection, calibration, and simulation of multiple links.

4.1.2 LOCUS 5G Testbed based on OAI

In this section, we describe how we implemented our 5G SDR-based experimental localization system. To overcome the excessive cost and size of traditional telecommunications equipment, we use a SDR approach that includes devices that can be easily accessed from the commercial market.

Experimental Setup: The experimental testbed consists of two customized workstations and two radio transceivers used to emulate the operator's network (5GCN and gNB) and the user terminal.

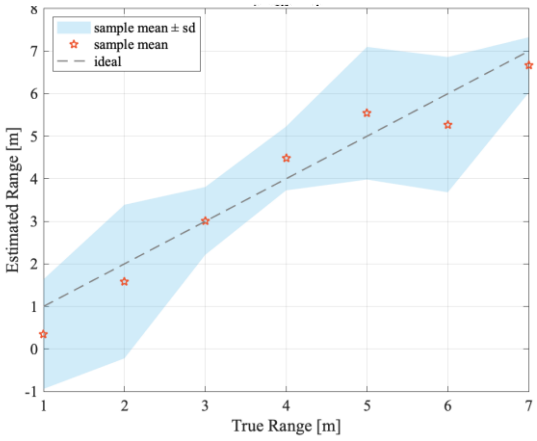


Figure 49: Estimated vs. true range obtained with calibrated data. The ideal bisector line is used for reference, the sample mean, and the standard deviation are illustrated varying the true range.

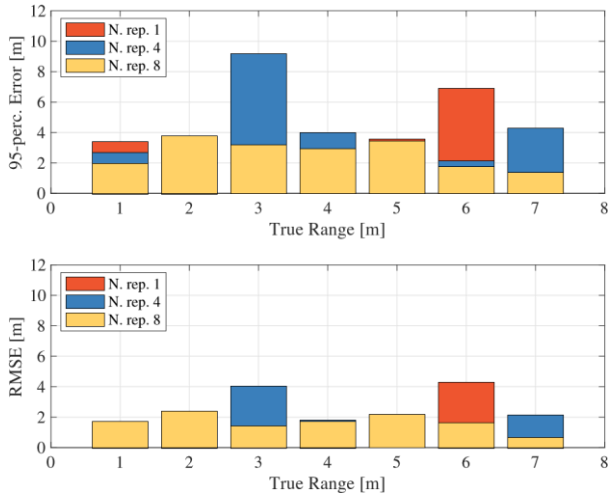


Figure 50: Ranging performance: top) 95-perc Error vs. True Range assuming PRS resource repetition factor = 1,4,8; bottom) RMSE vs. True Range assuming PRS resource repetition factor = 1,4,8.

Radio Transceiver — As radio front-end we used two USRP X310 devices [16] with UBX160 daughterboard. As they can be tuned over a wide radio frequency range, from 10MHz to 6GHz, they cover all the NR FR1 frequency bands with up to 160MHz of instantaneous bandwidth. By using the high-speed 10 GbE interface, we can achieve 184.32MS/s as sampling rate, thus theoretically achieving a localization accuracy of 1.627m. We used NR specific antennas attached to the SMA connectors of the SDR devices.

Workstation — For the localization experiments we used workstations powered by Intel Core i9 CPUs with 18 cores clocked at 3GHz. As OS, we used Ubuntu 18.04 LTS with kernel version 5.4.0-109-lowlatency.

Network SW platform — In our testbed we used OAI open-source project to implement Radio Access Network (gNB and 5G UE) and Core Network (5GC). OAI implements both DL PRS and UL SRS signals. We decided to focus on the DL PRS signal to validate and propose the tool, but it is possible to apply the same tool using the UL SRS signal as the positional signal. We used the most recent PRS dedicated develop commit available at the time of writing.

RF compatibility — We note here that we carried out all the experiments avoiding disturbing other UEs by either running tests inside the lab during weekends or stopping the experiments when other people were getting close to the lab. Considering the output power of the used SDR radios, we verified that the 5G NR signal emitted does not create interference outside the lab's testing area.

NR channel — In our experimental localization testbed we configured a 5G NR TDD channel in the N78 (3500MHz) frequency band with an instantaneous bandwidth of 80MHz (217 RBs) and 30KHz subcarrier spacing.

4.1.3 Dataset Collection

In this section we first present the experimental dataset collection, then the processing for data calibration, and finally the main results for ranging and positioning, following the processing scheme illustrated in Figure 48.

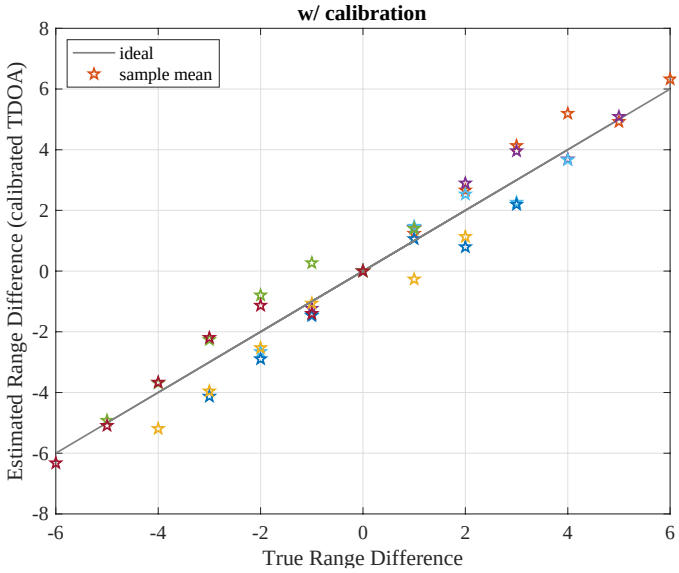


Figure 51: True vs. estimated range difference with calibrated data. The stars represent the sample mean for different datasets and are compared with the ideal bisector line.

We have collected a series of datasets containing the time of arrival measurements from the gNB to the user equipment as the distance varies. In particular, several datasets were obtained for d_{true} = 1,2,…,7m in LOS conditions. For each of these distances, we repeated the measurement as the number of repetitions N_{rep} = 1,2,…,8 changed. Fixed d_{true} and N_{rep} approximately 2000 measures of arrival time.

More specifically, when the UE receives the PRS signal from the gNB, it buffers the channel impulse response, applies the IDFT and estimates the ToA $t_i(d_{\text{true}},N_{\text{rep}})$ from the maximum peak of the channel impulse response in time.

The RSTD is calculated as the difference between the arrival times obtained by two gNBs. If the experimentation is carried out through two gNBs it will be necessary, that these are synchronized so that the signal is transmitted simultaneously. In the presence of a single gNB and to evaluate the localization process in the absence of synchronization errors, we calibrated the measurements to compensate any synchronization error at the transmission between gNBs. Considering this error as static, a calibration phase was carried out for each dataset. Note that this is a common approach in the presence of static timing error. The calibration considered the first 100 measurements of the dataset.

$$\tilde{t}_i(d_{\text{true}}, N_{\text{rep}}) = t_i(d_{\text{true}}, N_{\text{rep}}) - \frac{1}{N_{\text{cal}}} \sum_{i=1}^{N_{\text{cal}}} (t_i(d_{\text{true}}, N_{\text{rep}}) - d_{\text{true}}/c)$$

Note that such calibration eliminates any static source of error and that the measurements are all taken in LOS conditions.

Figure 49 shows the estimated range, i.e., $t_i(d_{\text{true}},N_{\text{rep}})c$ as a function of d_{true} with calibration. Figure 50 shows the RMSE and 95-perc. error on the range estimate varying the true range and the number of repetitions. It can be seen as the number of repetitions can truly reduce the range error and improve the TOA estimate. For example, for d_{true} = 6m the range error is below 7,2 and 1.8m in the 95% of the cases with 1,4, and 8 repetitions, respectively. Finally, we show the effect on the TDOA. Figure 51 shows the estimated range difference (e.g., the RSTD multiplied by the speed-of-light) as a function of

the true range difference varying the reference node and the dataset. In most of the cases, such difference is below 1 meter.

4.1.4 Simulation of the positioning system using experimental data

To evaluate the performance in terms of localization, a simulation based on the dataset of experimental measurements was carried out. The simulation of a system with N_g gNbs was performed considering a UE in central position and considering the gNB in position $[d_{1,i}\cos(\alpha_i), d_{1,i}\sin(\alpha_i)]$. In particular, the distance $d_{1,i}$ from the UE is uniformly chosen in the set $d_{\text{true}} \in \{1,2,\ldots,7\}$. As for the angle we have considered two configurations, with random α_i distributed between 0 and 360 degrees and with equal angular distance. For each implementation of the position estimation, we have chosen a random measure from the dataset $\tilde{t}_k(d_{1,i}, N_{\text{rep}})$ with k random, as N_{rep} varies.

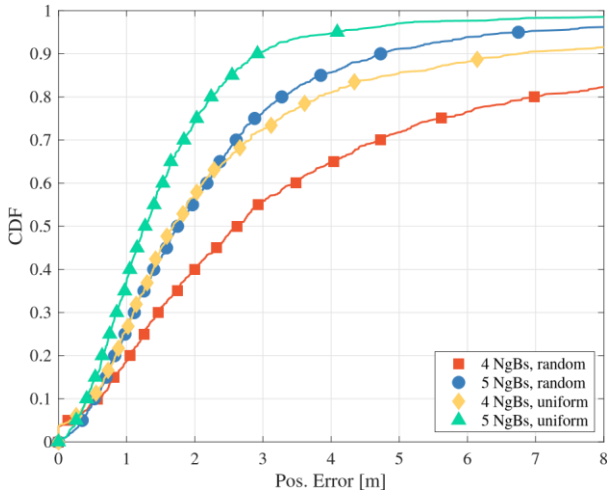


Figure 52: CDF of the positioning error varying the number of NgBs and using a uniform or random angular distance between them

Figure 52 shows the empirical CDF of the localization error obtained through linearized least square as the number of gNBs varies and using an equal or random angular distance between the various gNBs. The results therefore show the effect of the number of gNBs in the system as well as of the deployment of the network on the positioning performance.

5 Conclusion

This deliverable reported the integration work that was done to demonstrate the LOCUS technical achievements, through the evaluation of the platform, and the implementation of the project PoCs. Example results are showcased and demonstrate the added value of providing geolocation information to enable a variety of use-cases. It is important to highlight that the work in WP6 demonstrates not only the value of particular algorithms that are already thoroughly studied in other work packages, but also the added value of an architecture that (i) provides and end-to-end provisioning of localization analytics capabilities; (ii) can flexibly support stream-based real-time processing and database-based analytics; (iii) is cloud-native and integrates with the 5G ecosystem.

In the WP7 report on Innovation and Exploitation Plan and Actions [41], a further analysis of the potential industrial impact of LOCUS achievements is provided, in addition to a high-level SWOT analysis, and reporting of identified key innovations at the use-case level and at the platform level.

References

[1] LOCUS Project, "Deliverable 4.4: Implementation of the Virtualization platform for network control and management, final version." https://www.locus-project.eu/wp-content/uploads/2022/06/D4.4_nbm_29-4-22.pdf (accessed Nov. 04, 2022).

[2] LOCUS Project, "Deliverable 5.4: Prototype of the localization & analytics as a service solution." https://www.locus-project.eu/wp-content/uploads/2022/09/D5.4_4-8-22_nbm.pdf (accessed Nov. 04, 2022).

[3] "Consul by HashiCorp," *Consul by HashiCorp*. https://www.consul.io/ (accessed Nov. 04, 2022).

[4] LOCUS Project, "Deliverable 5.3: Design of the localization & analytics as a service solution." https://www.locus-project.eu/wp-content/uploads/2021/05/Deliverables-officially-submitted_D5.3_D5.3_11-5-21.pdf (accessed Nov. 04, 2022).

[5] "Swagger | API Documentation & Design Tools for Teams." https://swagger.io/ (accessed Nov. 04, 2022).

[6] "Apache Hive." https://hive.apache.org/ (accessed Nov. 04, 2022).

[7] "Apache Hadoop." https://hadoop.apache.org/ (accessed Nov. 04, 2022).

[8] LOCUS Project, "Deliverable 2.3: Security and Privacy, final version." https://www.locus-project.eu/wp-content/uploads/2021/05/Deliverables-officially-submitted_D2.3_D2.3_29-4-21.pdf (accessed Nov. 04, 2022).

[9] LOCUS Project, "Deliverable 3.1: 5G-based localization solutions, preliminary version."

[10] LOCUS Project, "Deliverable 3.2: 5G-based localization solutions, intermediate version."

[11] LOCUS Project, "Deliverable 3.3: Integrated localization technologies, intermediate version."

[12] LOCUS Project, "Deliverable 3.4: Integrated localization technologies, final version."

[13] LOCUS Project, "Deliverable 3.7: 5G-based localization solutions, final version."

[14] LOCUS Project, "Deliverable 6.2: Network management, network-assisted self-driving, people mobility and flow monitoring applications, integrated with geolocation mechanisms." https://www.locus-project.eu/wp-content/uploads/2022/03/D6.2_nbm_10-3-22.pdf (accessed Nov. 04, 2022).

[15] A. Conti, S. Mazuelas, S. Bartoletti, W. C. Lindsey, and M. Z. Win, "Soft Information for Localization-of-Things," *Proceedings of the IEEE*, vol. 107, no. 11, pp. 2240–2264, Nov. 2019, doi: 10.1109/JPROC.2019.2905854.

[16] S. Bartoletti *et al.*, "Location-Based Analytics in 5G and Beyond," *IEEE Communications Magazine*, vol. 59, no. 7, pp. 38–43, Jul. 2021, doi: 10.1109/MCOM.001.2001096.

[17] S. Bartoletti, Z. Liu, M. Z. Win, and A. Conti, "Device-Free Localization of Multiple Targets in Cluttered Environments," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 58, no. 5, pp. 3906–3923, Oct. 2022, doi: 10.1109/TAES.2021.3128964.

[18] LOCUS Project, "Deliverable 3.5: Integration with Device-Free Localization, preliminary version."

[19] LOCUS Project, “Deliverable 3.6: Integration with Device-Free Localization, intermediate version.”
[20] LOCUS Project, “Deliverable 3.8: Integration with Device-Free Localization, final version.”
[21] LOCUS Project, “Deliverable 4.1: Localization & Analytics for Smart Network Management, preliminary version.”
[22] “Juju | Operator lifecycle manager for K8s and traditional workloads.” https://juju.is/ (accessed Nov. 08, 2022).
[23] LOCUS Project, “Deliverable 2.5: System Architecture, final version.” https://www.locus-project.eu/wp-content/uploads/2021/12/D2.5_nbm_17-11-21.pdf (accessed Nov. 04, 2022).
[24] LOCUS Project, “Deliverable 4.3: Implementation of the Virtualization platform for network control and management, preliminary version.” https://www.locus-project.eu/wp-content/uploads/2021/05/Deliverables-officially-submitted_D4.3_LOCUS_D4.3_nbm_30-3-21.pdf (accessed Nov. 04, 2022).
[25] LOCUS Project, “Deliverable 4.5: Localization & Analytics for Smart Network Management, final version.”
[26] “Apache Airflow,” *Apache Airflow*. https://airflow.apache.org/ (accessed Nov. 11, 2022).
[27] LOCUS Project, “Deliverable 6.1: Detailed requirements from scenarios and application specification.” https://www.locus-project.eu/wp-content/uploads/2021/07/Deliverables-officially-submitted_D6.1_D6.1_27-6-21_nbm.pdf (accessed Nov. 04, 2022).
[28] A. Pinto, G. Santaromita, C. Fiandrino, D. Giustiniano, F. Esposito, and others, “Characterizing location management function performance in 5G core networks,” in *IEEE conference on network function virtualization and software defined networks*, 2022.
[29] F. Kaltenberger, A. P. Silva, A. Gosain, L. Wang, and T.-T. Nguyen, “OpenAirInterface: Democratizing innovation in the 5G Era,” *Computer Networks*, vol. 176, p. 107284, Jul. 2020, doi: 10.1016/j.comnet.2020.107284.
[30] W. G. van Toll, A. F. Cook IV, and R. Geraerts, “A navigation mesh for dynamic environments,” *Computer Animation and Virtual Worlds*, vol. 23, no. 6, pp. 535–546, 2012, doi: 10.1002/cav.1468.
[31] LOCUS Project, “Deliverable 5.1: Design and implementation of virtualization technologies and pattern recognition mechanisms for physical analytics, preliminary version.” https://www.locus-project.eu/wp-content/uploads/2021/02/Deliverables-officially-submitted_D5.1_D5.1-13-1-20.pdf (accessed Nov. 04, 2022).
[32] LOCUS Project, “Deliverable 5.2: Design and implementation of virtualization technologies and pattern recognition mechanisms for physical analytics, final version.” https://www.locus-project.eu/wp-content/uploads/2022/02/D5.2_nbm_4-2-22.pdf (accessed Nov. 04, 2022).
[33] G. Solmaz, J. Fürst, S. Aytaç, and F.-J. Wu, “Group-In: Group Inference from Wireless Traces of Mobile Devices,” in *2020 19th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, Apr. 2020. doi: 10/gjxwd9.

[34] “Main achievements – LOCUS Project.” https://www.locus-project.eu/results/main-achievements/ (accessed Nov. 07, 2022).
[35] “srsRAN.” srsRAN, Nov. 07, 2022. Accessed: Nov. 08, 2022. [Online]. Available: https://github.com/srsran/srsRAN
[36] “AETHER - Open Networking Foundation.” https://opennetworking.org/aether/ (accessed Nov. 08, 2022).
[37] “SD-RAN,” *Open Networking Foundation*. https://opennetworking.org/open-ran/ (accessed Nov. 08, 2022).
[38] “SD-Core,” *Open Networking Foundation*. https://opennetworking.org/sd-core/ (accessed Nov. 08, 2022).
[39] “Apache License, Version 2.0.” https://www.apache.org/licenses/LICENSE-2.0 (accessed Nov. 08, 2022).
[40] “onosproject.” https://docs.onosproject.org/ (accessed Nov. 08, 2022).
[41] LOCUS Project, “Deliverable 7.7: Report on Innovation and Exploitation Plan and Actions, v3.”