A Cognitive Serverless Framework for the Cloud-Edge Continuum

D5.5 Use Cases - Scientific Report - e

Version 1.0

30 April 2025

Abstract

COGNIT is an AI-Enabled Adaptive Serverless Framework for the Cognitive Cloud-Edge Continuum that enables the seamless, transparent, and trustworthy integration of data processing resources from providers and on-premises data centers in the cloud-edge continuum, and their automatic and intelligent adaptation to optimise where and how data is processed according to application requirements, changes in application demands and behaviour, and the operation of the infrastructure in terms of the main environmental sustainability metrics. This document provides an overall status of the contribution of the Project's software requirements towards meeting the user requirements that guide the development of the COGNIT Framework, offers additional information about the domains targeted by the Use Cases and the Partners involved in them, and provides an update on the Project's software integration process and infrastructure, on its testbed environment, and on the progress of the software requirement verification tasks during the Fourth Research & Innovation Cycle (M22-M27).

This project is funded by the European Union's Horizon Europe research and innovation programme under Grant Agreement 101092711 – SovereignEdge.Cognit

Deliverable Metadata

Project Title:	A Cognitive Serverless Framework for the Cloud-Edge Continuum
Project Acronym:	SovereignEdge.Cognit
Call:	HORIZON-CL4-2022-DATA-01-02
Grant Agreement:	101092711
WP number and Title:	WP5. Adaptive Serverless Framework Integration and Validation
Nature:	R: Report
Dissemination Level:	PU: Public
Version:	1.0
Contractual Date of Delivery:	31/03/2025
Actual Date of Delivery:	30/04/2025
Lead Author:	Thomas Ohlson Timoudas & Joel Höglund (RISE)
Authors:	Monowar Bhuyan (UMU), Malik Bouhou (CETIC), Aritz Brosa (Ikerlan), Idoia de la Iglesia (Ikerlan), Agnieszka Frąc (Atende), Torsten Hallmann (SUSE), Joel Höglund (RISE), Carlos Lopez (ACISA), Mateusz Kobak (Phoenix), Tomasz Korniluk (Phoenix), Antonio Lalaguna (ACISA), Fátima Fernández (Ikerlan), Carlos Lopez (ACISA), Philippe Massonet (CETIC), Nikolaos Matskanis (CETIC), Xavier Lessage (CETIC), Jean Lazarou (CETIC), Daniel Olsson (RISE), Mikel Irazola (Ikerlan), Álvaro Puente (Ikerlan), Holger Pfister (SUSE), Francesco Renzi (Nature 4.0), Marco Mancini (OpenNebula), Juan José Ruiz (ACISA), Kaja Swat (Phoenix), Paul Townend (UMU), Iván Valdés (Ikerlan), Yashwant Singh Patel (UMU), Riccardo Valentini (Nature 4.0), Pavel Czerny (OpenNebula), Filippo Tagliacarne (Nature 4.0).
Status:	Submitted

Document History

Version	Issue Date	Status[1]	Content and changes
0.1	22/04/2025	Draft	Initial Draft
0.2	28/04/2025	Peer-reviewed	Reviewed Version
1.0	30/04/2025	Submitted	Final Version

Peer Review History

Version	Peer Review Date	Reviewed By
0.2	25/04/2025	Antonio Álvarez (OpenNebula), Marco Mancini (OpenNebula)
0.2	28/04/2025	Torsten Hallmann (SUSE)

Summary of Changes from Previous Versions

First Version of Deliverable D5.5

[1] A deliverable can be in one of these stages: Draft, Peer-Reviewed, Submitted, and Approved.

Executive Summary

Deliverable D5.5, released at the end of the Fourth Research & Innovation Cycle (M27), is the fifth version of the Use Cases Scientific Report in WP5 ”Adaptive Serverless Framework Integration and Validation”.

This deliverable provides an incremental overview of the project's progress during the Fourth Research & Innovation Cycle (M22-M27). The document reports the implementation, tests and deployments done by the Use Case partners. An important shift during the reporting period to highlight is the transformation of the previous Use Case application work into the new project architecture. Additionally, it offers updates on the software integration process, infrastructure, testbed environment, and verification of software requirements.

The work reported in this document is complemented by the detailed project software resources reported in D5.8 COGNIT Framework - Software Source, and the project demo description presented in D5.11 COGNIT Framework - Demo. In turn, the document is complementary to the global overview provided through Deliverable D2.5 COGNIT Framework - Architecture, and the component-specific deliverables from WP3 (D3.4) and WP4 (D4.4).

The information in this report will be updated with a final stand-alone release at the end of the last remaining research and innovation cycle, i.e. M33.

Table of Contents

Abbreviations and Acronyms

5G	Fifth Generation Mobile Network
AD	Anomaly Detection
AI	Artificial Intelligence
API	Application Programming Interface
AUROC	Area under the receiver operating characteristic
AWS	Amazon Web Services
BDD	Behavioural Driven Development
CCAM	Cooperative, Connected and Automated Mobility
CC-CV	Current and Constant Voltage
CNN	Convolutional Neural Network
C-ITS	Cooperative Intelligent Transport System
DaaS	Data as a Service
DCC	Device Client in C
DDPG	Deep Deterministic Policy Gradient
DSRC	Dedicated Short Range Communications
DSO	Distribution System Operator
EC2	(Amazon) Elastic Compute Cloud
EN	European Norms
ENS	Esquema Nacional de Seguridad (Spanish National Security Schema)
ESS	Energy Storage System
ETSI	European Telecommunications Standards Institute
EV	Electric Vehicle
FaaS	Function as a Service
FAQ	Frequently Asked Questions
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GUI	Graphical User Interface
HEMS	Home Energy Management System
HTTP	Hypertext Transfer Protocol
HVAC	Heating, Ventilation and Air Conditioning
ICE	Infrastructure and Cloud Research & Test Environment at RISE
ID	Identifier
IP	Internet Protocol
IPv6	Internet Protocol version 6

ITS	Intelligent Transport System
kW	kiloWatt(s)
LTE	Long-Term Evolution
MAPEM	MAP (Topology) Extended Message
ML	Machine Learning
M-Hub	Mobility Hub (advanced TLC)
NB-IoT	NarrowBand - Internet of Things
OBU	On Board Unit
OCPP	Open Charge Point Protocol
OS	Operating System
PCI	Peripheral Component Interconnect
PPO	Proximal Policy Optimisation
PV	Photovoltaic
QA	Quality Assurance
QoS	Quality of Service
RES	Renewable Energy Source
REST	Representational State Transfer
RL	Reinforcement Learning
RSU	Road Side Unit
RTOS	Real-Time Operating System
SAE	Society of Automotive Engineers
SEM	Smart Energy Meter
SPATEM	Signal Phase And Timing Extended Message
SREM	Signal Request Extended Message
SSEM	Signal request Status Extended Message
SSH	Secure Shell
SSL	Secure Sockets Layer
SUMO	Simulation of Urban Mobility[2]
TCC	Traffic Control Center
TCP	Transmission Control Protocol
TEE	Trusted Execution Environment
TLC	Traffic Light Controller
TS	Technology Specifications
TSP	Traffic/Transit Signal Priority

[2] An open source, highly portable, microscopic and continuous multi-modal traffic simulation package designed to handle large networks: https://eclipse.dev/sumo/

TTC	TreeTalker Cyber
V2X	Vehicle to Everything communication technology
VM	Virtual Machine
VPN	Virtual Private Network

1. Introduction

This is the fifth version of the Use Cases Scientific Report. The initial version of the Use Cases Scientific Report (Deliverable D5.1), released in M3, provided an initial collection of user requirements, a description of each of the Use Cases—including an initial architecture design and a plan for the demonstration and validation to take place in Tasks T5.3, T5.4, T5.5, T5.6—and an update of the COGNIT Testbed environment.

Both the second and the third versions of the report (Deliverables D5.2 and D5.3), released in M9 and M15, provided summaries of the overall status of the contribution of the Project's software requirements towards meeting the user requirements that guide the development of the COGNIT Framework at the end of the First and Second Research & Innovation Cycle (M4-M9 and M10-M15). They also provided additional information about the domains targeted by the Use Cases and the Partners involved in them, listing new user requirements identified during the cycle, and offering an update on the Project's software integration process and infrastructure, on its testbed environment, and on the progress of the software requirement verification tasks per component.

Since Deliverable D5.3 there have been changes to the COGNIT architecture, motivated by the requirements of the Use Cases. This second version of the COGNIT Framework is described in Deliverable D2.4. During this Research & Innovation Cycle (M16-M21), the project partners have worked together with the Use Cases to transition to this new architecture, which, from the perspectives of the Use Cases, have required some modifications in how the Device Client interacts with the COGNIT Framework. D5.4 reported the status of the pilots at M21.

The present document, D5.5, gives an incremental update on the progress. It mostly follows the same structure as in previous versions, with the added inclusion of the Use Case Implementation and Deployment plans under each Use Case Section. D5.5 is composed of an introductory section and nine additional sections organised in two main blocks of content:

- Part I focuses on tracking the overall status (Section 2) and progress of the research and development work performed for each of the Use Cases, and their current status, with a dedicated section per Use Case (Sections 3 to 6).
- Part II focuses on the Project's software integration process and infrastructure (Section 7), on the evolution of the testbed environment (Section 8), and on the software requirements verification tasks carried out per component (Section 9).

The document ends with a brief conclusion and a next steps section (Section 10).

PART I. Validation Use Cases

2. Overall Status

The table below shows the current status of each Software Requirement towards meeting its associated global and user requirements, following a simple colour code: for activities that have not started yet, for activities in progress, and for completed activities:

	ID	DESCRIPTION	Device Client						COGNIT Frontend	Edge Cluster		Cloud-Edge Manager						AI-Enabled Orchestrator		Secure & Trusted Execution of Computing Environments		
			SR1.1	SR1.2	SR1.3	SR1.4	SR1.5	SR1.6	SR2.1	SR3.1	SR3.2	SR4.1	SR4.2	SR4.3	SR4.4	SR4.5	SR4.6	SR5.1	SR5.2	SR6.1	SR6.2	SR6.3
Sovereignty	SOR0.1	The COGNIT Framework shall be able to leverage public, private, and self-hosted cloud and edge infrastructures hosted in the European Union.																				
	SOR0.2	The implementation of the COGNIT Architecture shall maximise the use of European open source technologies and frameworks.																				
	SOR0.3	The COGNIT Framework shall provide an abstraction layer that ensures workload portability seamlessly across different infrastructure providers.																				
	SOR0.4	Data handling by the COGNIT Framework shall be compliant with the GDPR.																				
Sustainability	SUR0.1	Sustainability performance needs to be measurable (e.g. energy profiles should be queryable and updatable for every feature/component within the framework), including energy sources (e.g. renewable, non-renewable) and energy consumption profiles (e.g. estimated power consumption).																				
	SUR0.2	Sustainability needs to be maximised to reduce environmental footprint by leveraging edge characteristics (e.g. by increasing the share of renewables, minimising battery use/size, using energy otherwise wasted, or scaling down active Runtimes).																				
	SUR0.3	The whole energy lifecycle should be taken into account in order to implement a circular economy, including e.g. energy availability and cost and hardware degradation.																				
Interoperability	IR0.1	Deployment of the COGNIT Framework and of its components should be as portable as possible across heterogeneous infrastructures or cloud/edge service providers (e.g. by using broadly-adopted virtualisation and container technologies).																				
	IR0.2	Preference should be given to expanding existing frameworks, tools, and open standards.																				
	IR0.3	The interfaces of the COGNIT Framework shall be documented in order to facilitate discovery of its features by third-parties.																				
Security	SER0.1	Communications inside COGNIT, and between the COGNIT environment and the outside (e.g. IoT devices) should be encrypted and signed using security mechanisms such as SSLv3.																				
	SER0.2	The COGNIT Framework should be built following security-by-design and Zero Trust practices.																				
	SER0.3	The implementation of the COGNIT Framework should be aligned with the latest legislative frameworks, such as the NIS2 Directive, the GDPR, and the future Cyber Resilience Act (CRA).																				
	SER0.4	Runtimes should be protected against threats by the enforcement of security controls such as secure defaults, vulnerability scans, intrusion and anomaly detection and continuous security assessment (the specific controls to be implemented will be determined by a risk analysis).																				
	SER0.5	Resources should be protected by an Identity and Access Management (IAM) system, implementing role based access control (RBAC), security zones, and support for a multi-tenant security model.																				
	SER0.6	Integrity of the offloaded functions needs to be guaranteed, including the function inputs and outputs (also during the live migration of FaaS Runtimes).																				

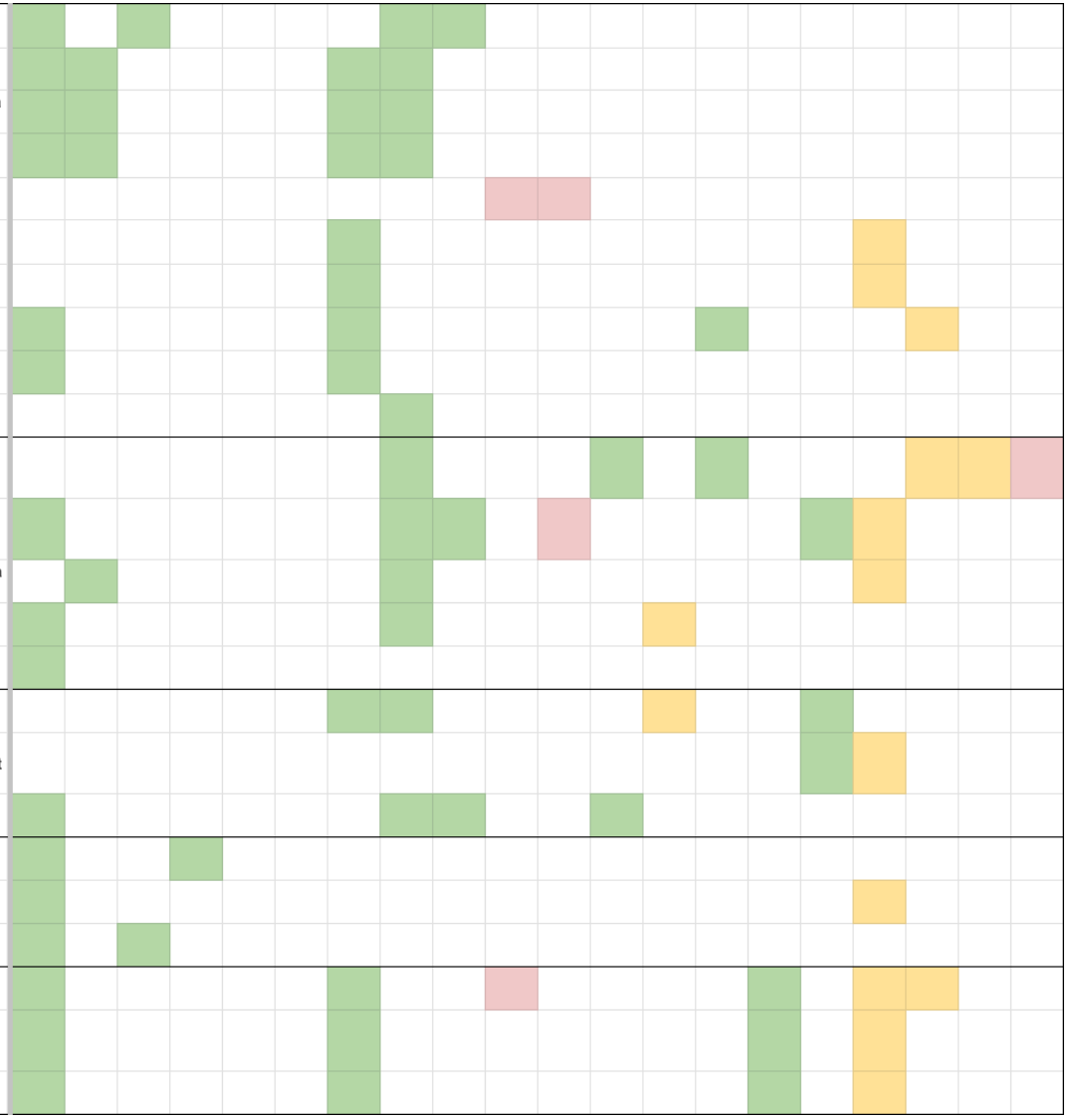
Common Requirements	UR0.1	Device applications should be able to offload any function written in C or Python languages.	
	UR0.2	Device applications should be able to upload data from the device ensuring data locality with respect to where the offloaded function is executed.	
	UR0.3	Device applications should be able to upload data from external backend storages ensuring data locality with respect to where the offloaded function is executed.	
	UR0.4	Execution of functions such as ML inference engines should be able to load machine learning models stored ensuring data locality with respect to where the function is executed.	
	UR0.5	Function execution can be executed in different tiers of the Cloud-Edge continuum according to network latency requirements.	
	UR0.6	Device application shall have the ability to define maximum execution time of the offloaded function upon offloading.	
	UR0.7	Device application shall have the ability to specify and enforce runtime maximum provisioning time and runtime shall be provisioned within the previously specified time.	
	UR0.8	Device applications must be able to request and obtain an authorization prior to establishing any further interaction with COGNIT.	
	UR0.9	IAM system integration for high granularity authentication and user management for device clients, provisioning engine and serverless runtime.	
	UR0.10	Push mechanism to inform about status or events from the COGNIT Framework back to the requestor device client.	
UC1	UR1.1	Function execution shall be supported in shared, multi-provider environments (with different access and authorization procedures), and the execution must be isolated from other processes on the host system.	
	UR1.2	Device application shall have the ability to dynamically scale resources for offloading function execution to maximise exploitation of resources in shared environments, while avoid saturation or resources kidnapping.	
	UR1.3	Function execution should exploit data locality and prioritise edge nodes where the required data is already stored.	
	UR1.4	The whole life cycle of either function execution or code offloading should be auditable and non repudiable.	
	UR1.5	Device applications should be able to request execution over GPUs.	
UC2	UR2.1	It shall be possible to obtain both a-priori estimates of expected, and actual measurements of, energy consumption of the execution of function.	
	UR2.2	COGNIT Framework should be able to adapt to rare events with sudden peaks of FaaS requests, in which the offloaded function requires much heavier computations and more frequent execution than usual.	
	UR2.3	Possibility for devices to request access to GPUs, when available, during high-alert mode.	
UC3	UR3.1	Device Client and user applications shall share a maximum of 500 kB of available RAM in total.	
	UR3.2	It shall be possible for the user application to dynamically scale up/down resources for function execution due to changes in the user preferences.	
	UR3.3	The SDK for the Device Client shall have support for the C programming language.	
UC4	UR4.1	The Device Client should have the ability to dynamically set the permissible edge nodes for executing the function based on policy (e.g. geographic security zones, distance to edge node).	
	UR4.2	The COGNIT Framework should have the ability to live migrate of data/runtime to different edge locations based on policy and location of function execution (e.g. geographic security zones, distance to edge node).	
	UR4.3	The Device should be able to request the execution of a function as close as possible (in terms of latency) to the Device's location.	

Table 2.1. Current status of each Software Requirement towards meeting its associated global/user requirements.

3. Use Case #1: Smart Cities

3.1 Implementation and Deployment Plan M22-M27

This is the Implementation and Deployment Plan made and executed by the Smart Cities partners during the reporting period.

Implementation and deployment plan

- Setting up a COGNIT Edge Cluster on premise as a testbed for UC1.
- Installing and setting up all the M-Hubs to be operative and ready to start testing with the COGNIT Edge Cluster.
- Creating a junction traffic model for simulation, of one of Granada's pilot junctions.
- Creating Serverless Runtime performing the following basic functionality:
    - Making a request to the simulator passing the simulation parameters and getting the resulting KPIs.
- Identifying usage patterns, i.e. frequency of bus detections on junctions of the scope.
- Creating mock requests using performance monitoring tools to simulate FaaS requests made by M-Hub.
- Deploy a COGNIT Edge Cluster in ACISA's premises in Barcelona with only one node, prepare for a second node in order to test load balancing, node failure, etc.

Validation criteria

- C-ITS Service "Traffic Signal Priority" (TSP) is tested in the pilot intersections, and basic functionality is validated in Granada.
- A junction is modelled as a Digital Twin, and is ready to run simulations and obtain results in the form of KPIs.
    - TSP, enhanced with a Digital Twin model is tested and basic functionality is validated. Edge node is able to launch the requested simulations by the FaaS.

3.2 Use Case Scenario and Architecture

A major improvement in this cycle has been the deployment of a COGNIT Edge Cluster Frontend and the end to end integration with the COGNIT Frontend and the main testbed at RISE ICE , and enabling the COGNIT services in this new cluster.

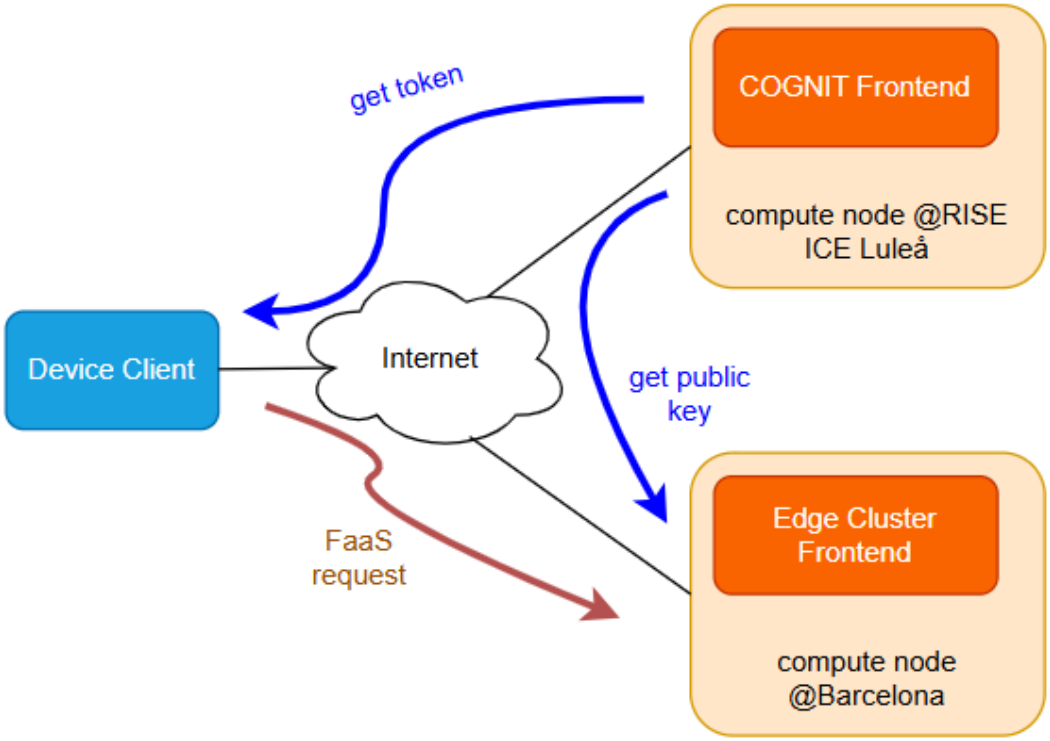


Figure 3.1: General architecture for direct FaaS request to Edge Cluster

In this scenario, until the AI-Enabled Orchestrator has been fully integrated and operational, the destination of the FaaS is hardcoded so that the requests can reach the desired Edge Cluster in Barcelona for our testing purposes.

Currently, the working assumptions regarding the constraints for the function execution are that the response time should not exceed 1 second. Here, response time means the elapsed time from the client perspective between submitting a function request to receiving the function result. The function result is not operation critical, in the sense that the system will function without it, but beyond that time frame the result quickly loses relevance (since the decision must be made in a matter of seconds) and will negatively impact traffic flow optimisations and service level agreements.

In this development cycle, we added new KPIs to the Sumo model so that the values returned are better suited for the decision process at the M-Hub about granting or not the priority requests.

The integration of the device client with the M-Hub device will be made during the next development cycle. In the current state, we are generating FaaS requests using a mock program or a performance reporting tool.

3.3 Summary of developments during cycle M22-M27

In this development cycle we made progress in the following areas:

- Modify the Smart Cities Device Client software to adapt it to the new architecture V2.0.

- Tests of FaaS execution using this new architecture and device client framework software were made successfully. The FaaS has been executed in the compute nodes at ICE[3] Edge Cluster.
- Deployment of local COGNIT Edge Cluster in Barcelona.
- Tests of FaaS execution in the Edge Cluster at Barcelona. Some manual adjustments in the device client framework were needed to redirect the requests to the Edge Cluster in Barcelona.
- New iteration of the Sumo model for one of the Junctions in Granada, to add a new KPI closely related to our actual needs about Priority simulation results. The simulation model of the intersection 1001 has been created. Traffic light functionality and traffic demand have been added to be used in the test scenario.
- Usage pattern identification: Tests have been conducted using a preliminary traffic simulator, without the use of COGNIT, so that the bus priority level is determined based on simulation results of behaviour patterns, which are then submitted to the controller. These experiments have helped refine the algorithm that will be implemented in the FaaS.
- Tests and validation of the TSP in Granada: Once the system was launched, statistics on bus detections were collected, and various tests were carried out in order to find the optimal way to manage priorities. This acquired knowledge will be applied to the final development of the FaaS.
- Digital twin modelling: The digital model of the intersection requires modifications at the edge and the creation of models within the SATURNO platform. These models have been defined and implemented in both components and will serve as the basis for obtaining operational parameters by the FaaS.
- Completed testing (excluding FaaS functionality) and data gathering for the V2X Public Transport Prioritisation (PTP) system in Granada, consisting of 38 Mobility Hubs managing 114 intersections.
- Migration of the Saturno platform to new facilities, ensuring its integrity.

3.4 Integration with the COGNIT Framework

Application structure

The integration of the device client with the M-Hub device will be made during the next development cycle; hence in the current state we are generating FaaS requests using a mock program or a performance reporting tool.

Integration with the COGNIT Framework

In the current development cycle, the COGNIT Edge Cluster Frontend has been deployed in the cluster in Barcelona. This involved the following tasks:

- Deployment of the KVM hypervisor on the compute node of the Edge Cluster.
- Integration with VPN of Testbed Environment at RISE ICE.

[3] https://www.ri.se/en/ice-datacenter

- Enable cross-authentication between the Cloud-Edge Manager and the new Edge Cluster.
- Integration of the Edge Cluster with the Cloud Edge Manager.
- Creation of a cluster and a virtual network in the Cloud Edge Manager.
- Configuration of Service and VM templates for the Serverless Runtime.
- Deployment of the COGNIT Edge Cluster Frontend.
- Networking configurations to enable two-way communication between the Edge Cluster node and the VMs, access of VMs to the Internet to allow deployment of libraries and dependencies, and inbound connection from a public IP to the VM running the COGNIT Edge Cluster Frontend in the Edge Cluster.

Use Case Internal Testbed

The Use Case has set up a testbed in Granada for testing the integration of the real device in the field, with V2X sensors to receive buses or emergency vehicle detections, and running a simplistic algorithm to make decisions about priority. This algorithm will be replaced with a more elaborate one using COGNIT FaaS requests to run simulations in the upcoming cycles.

Furthermore, the following hardware has been acquired and is being deployed in Barcelona to provide local processing power to the pilot. In the current development cycle, one of the nodes has been used to deploy a COGNIT Edge Cluster Frontend:

1x Supermicro Twin rackable SuperServer SYS-120TP-DTTR Chassis 2U

Representing two nodes, each one with these characteristics:

- 2 x CPU Intel Xeon Silver 4314 16C/32T 2.4GHz.
- 128GB RAM: 8 modules x16GB DDR4-3200.
- 2 x Discs SSD de 960GB <2DWPD.
- 2 x Ports 10Gb Base-T @ motherboard.
- 2 x Ports 1Gb Base-T @ an additional board.

1x Rackable server 2U supermicro

- 2 x CPUs Intel Xeon Silver ICX 4314 16C/32T 2.4GHz.
- 128GB RAM: 8 modules x 16GB DDR4-3200.
- 2 x Discs SSD de 480GB SATA Intel D3 S4520 < 2DWPD.
- 2 x Ports 10Gb Base-T @ motherboard.
- NVIDIA QuadroRTXA5000 24GB GDDR6.

Saturno platform, previously deployed on outdated servers, has been migrated to a new cluster to ensure its integrity and availability:

4x Rackable DELL server DELL PowerEdge R660

- 2 x CPU Intel Xeon Silver 4416 20C/40T 2.0GHz.
- 512GB RAM: 16 modules x32GB 5600MT.
- 2 x Discs SSD de 480GB.
- 2 x Ports 10Gb Base-T @ motherboard.

- 2 x Ports 1Gb Base-T @ an additional board

Figure 3.2 shows the devices installed in each intersection of the Granada pilot:

Figure 3.2. Devices deployed on the pilot in Granada.

Figure 3.3 illustrates a map of UC1 pilot testbed, which utilized the infrastructure of Granada's public transport Line 4. This system comprises 114 intersections managed by 38 M-Hubs and 38 RSUs, along with 20 public buses equipped with V2X technology:

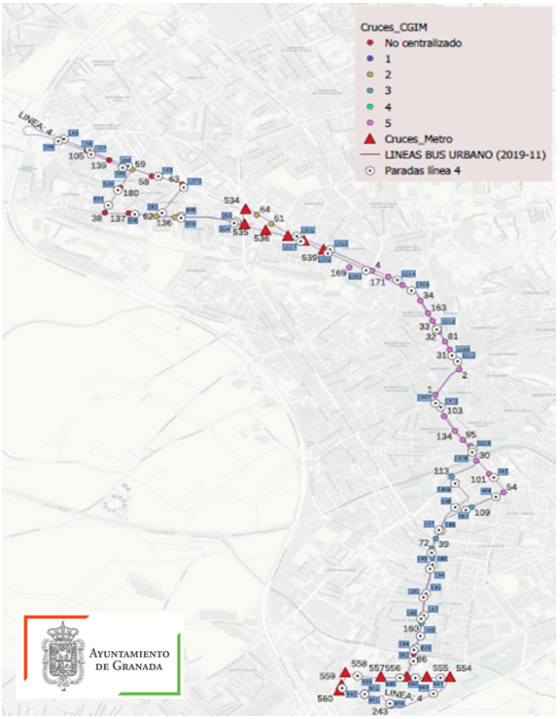


Figure 3.3. Map of Granada's public transport Line 4, including intersections and bus stops.

Project Demo Use Case Testbed

Figure 3.4 presents the elements of the pilot on each segment (Vehicle or Control Center), including the ETSI standardized messages, V2X subsystems, M-Hub, Saturno, and the COGNIT Framework:

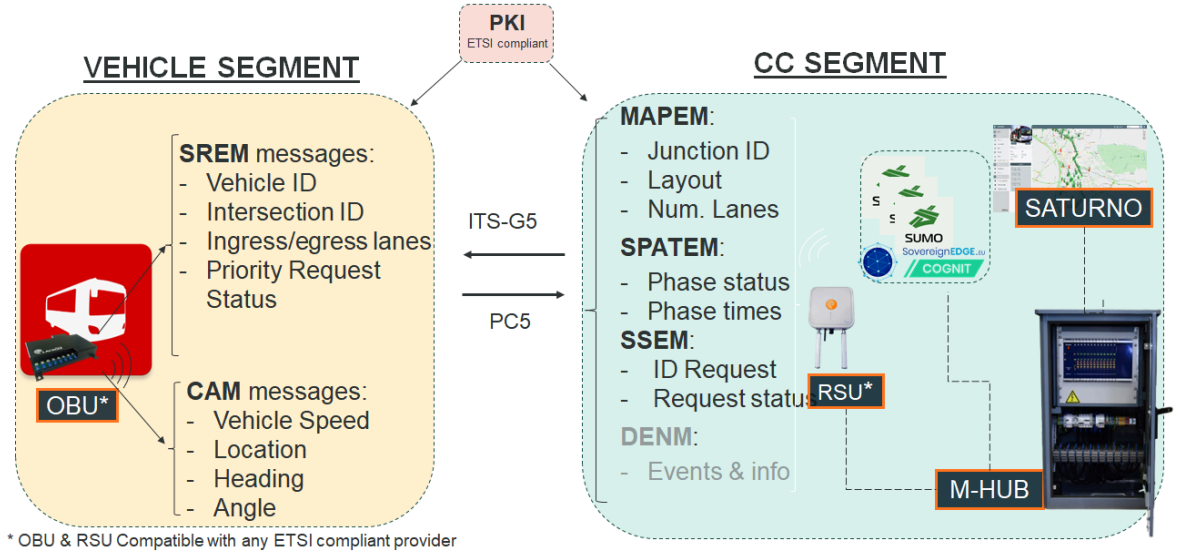


Figure 3.4. Vehicle Segment and Control Center (CC) Segment, showing V2X messages and devices installed in each intersection.

Figure 3.5 shows some preliminary results from the PTP system in several intersections. The service was tested in two scenarios: with and without the priority system enabled. The graph demonstrates that average transit time is reduced significantly when the priority system is active (indicated by green):

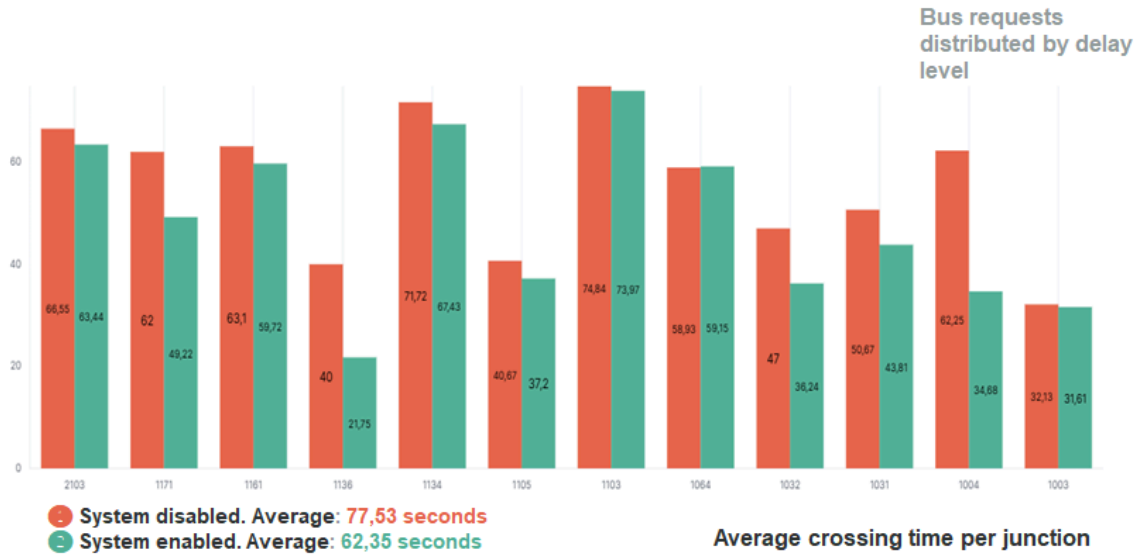


Figure 3.5. Preliminary experimental results of the V2X PTP system.

Note: we are using a specification based upon **Behavioural Driven Development**(BDD)

A description of each specific scenario of the narrative with the following structure:

Given: the initial context at the beginning of the scenario, in one or more clauses;

When: the event that triggers the scenario;

Then: the expected outcome, in one or more clauses.

Source: https://en.wikipedia.org/wiki/Behavior-driven_development

We are considering the following scenarios:

Scenario - 1

Given I configure the environment for a flavour Smartcity-ice.

When a FaaS request is sent from a PC.

Then, a response of KPI for priority is received at the device client.

And the FaaS is run in the ICE Edge Cluster.

Scenario - 2

Given I configure the environment for a flavour Smartcity-bcn.

When a FaaS request is sent from a PC.

Then, a response of KPI for priority is received at the device client.

And the FaaS is run in the BCN Edge Cluster.

Scenario - 3

Given I upload simulation results data to the DaaS belonging to a certain range of time.

When a FaaS request is sent from a PC that involves simulations of that range of time.

Then, a response of KPI for priority is received at the Device Client.

And the FaaS is using the result data available in the DaaS.

Scenario - 4

Given I upload simulation results data to the DaaS belonging to a certain range of time.

When a FaaS request is sent from a PC that involves simulations of a range of time not available in the DaaS.

Then, a response of KPI for priority is received at the device client.

And the FaaS is running the simulation for that range of time.

And the result of the simulation for this range of time is stored on the DaaS.

Scenario - 5

Given I configure a M-Hub environment for a flavour Smartcity-bcn.

When a bus approaches a detector.

Then, a FaaS request is sent.

And the FaaS is run in the BCN Edge Cluster.

And a response of KPI for priority is received at the M-Hub.

4. Use Case #2: Wildfire Detection

4.1 Implementation and Deployment Plan M22-M27

This is the Implementation and Deployment Plan made and executed by the Wildfire Detection partners during the reporting period.

Implementation plan

- Adapt all previous work to the new architecture:
  - Migrate the UC2 Serverless Runtime flavour with the fire recognition algorithm to the new architecture.
  - Test that the function offloading works correctly in the new architecture.
- Finalize two wildfire models to test the response of COGNIT Framework to sudden peaks of requests and adapt them to the new architecture.
- Finalize the first TreeTalker Cyber Fire prototype.
- Send a FaaS request from the prototype using the C Device Client or writing a specific piece of code compatible with its Internet module following COGNIT documentation.

Validation Criteria

- The simulators are ready.
- The image recognition function can be offloaded by the simulators.
- The prototype can offload a function to the COGNIT Framework.

4.2 Use Case Scenario and Architecture

The use case scenario and architecture are described in detail in the previous deliverable “D5.3 Use Cases - Scientific Report - c”, within section 4.1 “Current architecture and scenario” and were updated in “D5.4 Use Cases - Scientific Report - d”.

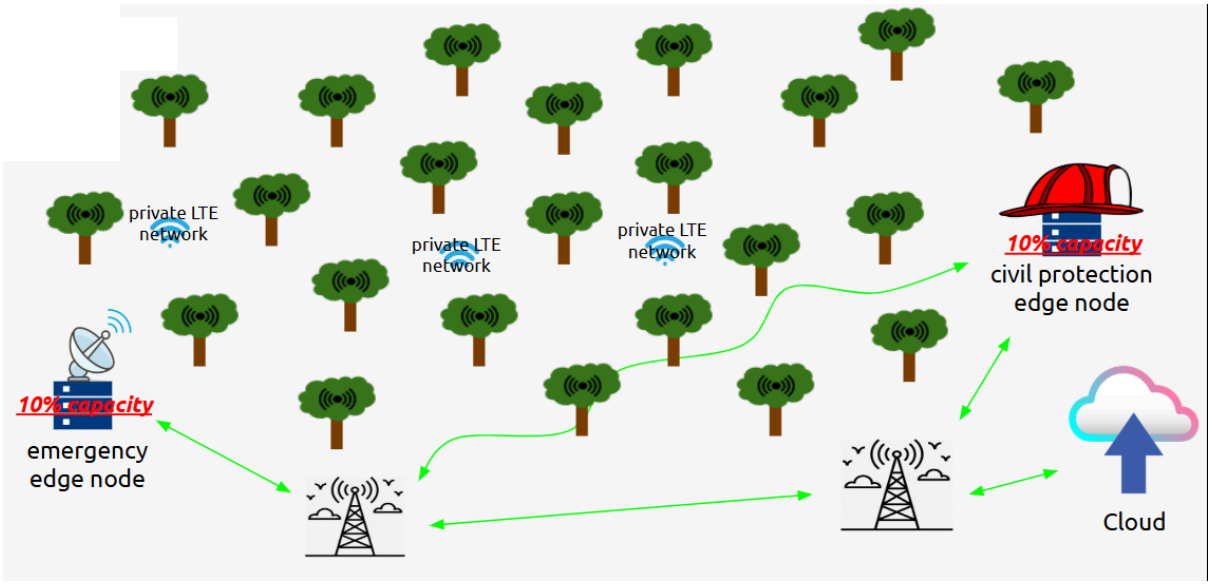


Figure 4.1: Example of a possible implementation of the wildfire early detection network

The UC2 architecture, as shown in Figure 4.1, has not changed substantially from the previous cycle.

Currently, the working assumptions regarding the constraints for the function execution would mainly concern latency, renewable energy usage, and costs, although the specifics about these constraints are under investigation. One potential trade-off to consider is between costs and renewable energy usage, where the user might want to maximize renewable energy utilisation given a maximum cost.

4.3 Summary of developments during cycle M22-M27

During the development cycle M22–M27, efforts focused on updating the previous work to align with the new architecture while simultaneously developing the demo and application for use in the next cycle. Both the UC2 Serverless Runtime image and the Python function, along with the related software, were updated and tested during this phase. The function was successfully offloaded to COGNIT Framework Version 2 and returned the expected results.

A first version of the C client was released, along with a Serverless Runtime capable of executing functions based on C client requests. The UC flavour on COGNIT testbed was updated to the new Serverless Runtime (development version), and tests confirmed that the C client could offload functions to it. Additionally, discussions were held regarding future steps to adapt the device C client to the needs of the TreeTalker Cyber Fire project.

The device itself is under development, and some components have been tested to improve its flame detection feature. Specifically, a 3MP ArduCam was acquired to improve image resolution, and an MLX90640 sensor was purchased to potentially replace the existing UV radiation sensor. The MLX90640 consists of an infrared camera (32x24 resolution) that provides the average temperature for each pixel, offering greater precision in flame detection compared to a single-point sensor. Several gas sensor models were selected, with some already tested and others still being acquired. Additionally, a 4G module (Simcom A7268E) was tested as an alternative in case NB-IoT technology proves unsuitable for image offloading.

The wildfire behaviour in the spatial simulation was improved by approximating fire spread using ellipses and incorporating wind angle effects (Figure 4.2). Currently, the statistical model is being updated to COGNIT Framework Version 2, with parallelisation added to better simulate real-world scenarios. The ongoing simulations will also be used for the final demo.

Figure 4.2: Images of spatial simulation at different times

4.4 Integration with the COGNIT Framework

Currently, UC2 is working on the finalization of the device; the integration of a camera managed by an esp32 with the selected processor, as well as the Internet connection by NB-IoT technology were already tested (Figure 4.3). The work is focusing on improving the resolution of the collected images as well as implementing the remaining gas sensors in order to have the prototype ready.

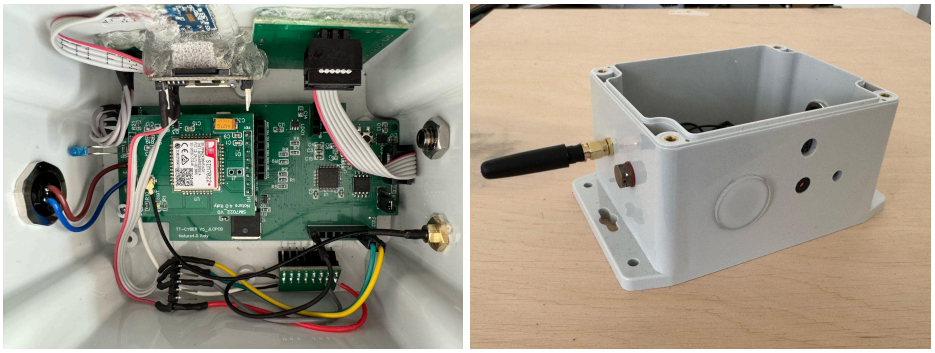


Figure 4.3: First prototype of TreeTalker fire with internet connection and ESP32-CAM

The function to offload has already been tested and updated to be compatible with COGNIT Framework Version 2 and can be successfully offloaded. The two demos to simulate the simplified spreading of a fire are currently being polished; the statistical simulation was updated to offload the function in parallel in order to simulate the pattern of the requests the network could offload during a wildfire, as well as the number of concurrent requests.

Additionally an Edge Cluster with one node has been set up during the previous cycle to test the possibilities to offload the function to a local Edge Cluster as well as the integration of new Edge Cluster to an existing COGNIT deployment. It is particularly useful since Nature 4.0 is interested in deploying COGNIT in private networks of civil protection organisations that manage their private servers and compute node resources due to the sensitivity of the data and for security reasons. The compute node of the Edge Cluster in Nature 4.0 headquarters must be updated with the latest changes.

Overall, the UC2 is currently working on finalising all the components that will be used to demonstrate the successful integration with the COGNIT Framework.

Application structure

The application consists of a network of devices that periodically scan their environment for signs of fire. These devices act as clients, offloading an image recognition function to the COGNIT Framework. The frequency and number of concurrent requests increase significantly during high-alert mode, which is triggered by wildfire events. The offloaded function utilises a convolutional neural network (CNN) to detect forest fires. If a fire is confirmed by the algorithm, additional devices in the area are alerted.

Every device that detects the presence of a flame increases its sampling frequency in order to provide better insight into the situation. The function was selected in previous cycles of the project and can be successfully offloaded to COGNIT Framework Version 2 targeting UC2_V2 flavour. Geographically distributed Edge Clusters play a crucial role in enhancing network reliability by mitigating risks associated with network instabilities during wildfire events, thereby reducing potential points of failure. A more detailed description of the UC2 application can be found in the deliverable “D5.4 Use Cases - Scientific Report - d”, in the “Application Structure” section.

Integration with the COGNIT Framework

The UC2_V2 flavour is up to date and can run the image recognition function that has also been updated in this cycle to work with COGNIT Framework Version 2.

An Edge Cluster has been deployed in Nature 4.0 to test the integration of the Edge Cluster with the central testbed in RISE. The integration will be performed in the next cycle.

Use Case Internal Testbed

The UC2 internal testbed is composed of the prototype and the edge node.
The device is under development, with the target being a platform capable of collecting the following parameters:

- Carbon dioxide concentration in air (ppm).
- Ozone concentration in air (ppm).
- Particulate matter PM10, PM2.5, PM1 (μg/m^3).
- Air temperature (°C).
- Air relative humidity (%).
- Sensor with an IR radiation matrix for flame detection.
- Camera images.
- Geographic coordinates hardcoded in the device.

The system components have been adapted based on test results to optimise fire detection and hardware performance. The processor selected in the previous cycle was the STM32L433RCT6, but the use of a Raspberry Pi Zero or an esp32 as the main processor is currently under evaluation. Adopting the Raspberry Pi would also enable running Python-based software. The ESP32 + OV2640 camera solution was implemented, but the ArduCam is considered a better option in terms of resolution, despite requiring more

memory. Additionally, it can be directly connected to the processor. An IR camera has also been tested as an alternative to the single-point solution (Figure 4.4). For internet connectivity, an NB-IoT module was developed and tested, but during this cycle, a 4G module was also introduced and tested with the Raspberry Pi to provide higher transfer speeds if needed.

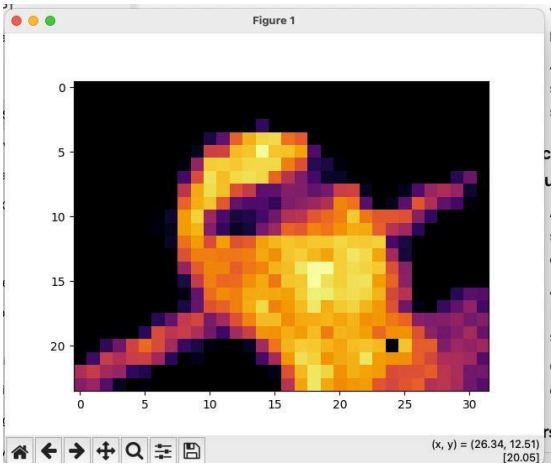


Figure 4.4: Infrared camera image captured during tests

The ESP-CAM and NB-IoT connection were successfully tested in the previous cycle. However, the device continues to be improved to achieve the best possible prototype by the end of the project.

Moreover, a node within the Edge Cluster was deployed in the previous cycle in Nature 4.0 headquarters with the following characteristics:

- 2 x CORSAIR VENGEANCE LPX DDR4 RAM 64GB (2x32GB) 3200MHz.
- 2 x SAMSUNG MZ-77E4T0B/EU 870 EVO SSD 4 TB.
- 2 x Crucial P3 Plus SSD 2TB PCIe Gen4 NVMe M.2 SSD.
- 1 x AMD Ryzen Threadripper PRO 5975WX processor 3.6 GHz 128 MB L3.
- 1 x GeForce RTX® 4090 24GB.

It will be connected to the RISE main testbed in the following cycle.

Project Demo Use Case Testbed

The demo will be composed of three different elements:

- At least one working prototype able to offload the function to the COGNIT Framework, possibly using the C Device Client;
- One or both simulations to reproduce the behaviour of the device during wildfire events and show the features of COGNIT, in particular its scalability during unforeseen peaks of requests;
- An Edge Cluster to test its integration and use with the main testbed.

The Edge Cluster will be connected to the project testbed in RISE ICE, Luleå, to demonstrate the behaviour of the framework and the interaction of the application with an Edge Cluster. The wildfire use case is actively working on the development and finalisation of the demo's elements. The possibility of a pilot is still under evaluation, depending on the final results obtained.

5. Use Case #3: Energy

5.1 Implementation and Deployment Plan M22-M27

This is the Implementation and Deployment Plan made and executed by the Energy partners during the reporting period.

Basic Implementation Plan

- Setup a COGNIT Edge Cluster on premise as a testbed for UC3. In progress.
- Prepare a dockerized image for running multiple instances of UC3 simulation for testing the scalability of the COGNIT Framework. Done.
- Run Device Client in C on the electricity meter. Done.
- Implement a stable AI decision algorithm. Done.

5.2 Use Case Scenario and Architecture

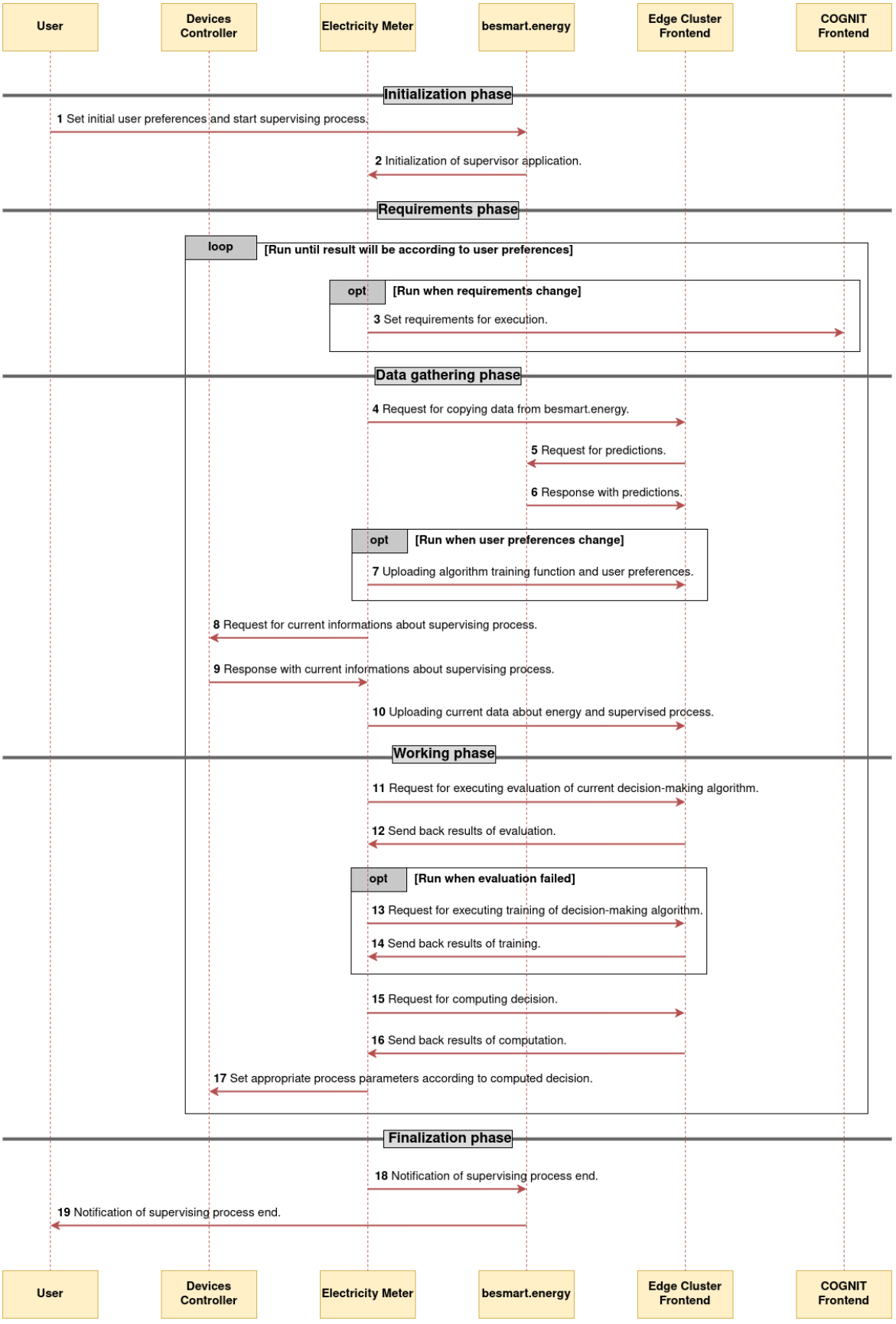


Figure 5.1. Process diagram for a single energy meter.

The Use Case scenario has been extensively described throughout previous deliverables (see D5.3 and D5.4); this deliverable only describes the changes that have been made.

The new part of the reference scenario is the introduction of online training of the AI model. Using the output returned from the trained model, a decision is made regarding the control of end devices, and thus the energy flow in the household grid. The new offloaded function differs in its performance requirements as it takes about 7-8 minutes to run and requires more computational resources. According to the scheme presented in Figure 5.1, the request to execute this function is made based on the results of the evaluation function, which will be the third type of function executed on COGNIT per single energy meter. As we are still working on proper metrics to evaluate the model, for now, the training is offloaded periodically, for example, after every 24h of simulation, to ensure the model is adapted to the newest data.

Currently, the working assumptions regarding the constraints for the function execution are that the response time should not exceed 1 minute. Here, response time means the elapsed time from the client perspective between submitting a function request to receiving the function result. Since the overall goal of this use case is maximisation of locally produced green energy usage, there could be interest in using green energy also for the executing of the function. While this objective is well-aligned with the flexibility afforded by the relatively relaxed response time constraints, there could be some further constraints about the location for execution because of the need for locality and confidentiality of sensitive data.

5.3 Summary of developments during cycle M22-M27

AI Algorithm

During this cycle, research regarding an AI-based solution for home energy management systems (HEMS) was focused on stabilising model training. Our aim was to ensure that, regardless of the nature of the input data, the geographic location of the meter and the settings of the end devices in the household, every training would lead to getting a model capable of managing a home network. Reinforcement learning was proven in literature to be successful in similar cases, so we tested our previously implemented training algorithm on a larger range of energy data. We introduced a few improvements like scaling of inputs, updating parameters after a batch of epochs and adding noise directly to the action vector. Following the most up-to-date articles in this field, we changed the learning method to Proximal Policy Optimisation (PPO). In PPO the agent is composed of critic and actor modules. The actor’s objective is to determine the optimal policy, considering the environment, to maximise the reward. Therefore, the actor module is responsible for generating the action of the system, represented by the policy. The critic module estimates the value function of the state of the system. To estimate the expected cumulative reward, the critical module uses the Q-value function. The critic’s output value is used by the actor to adjust policy decisions, leading to a better return. One of the advantages of the PPO is that its formulation helps to maximise exploration in the learning process without increasing the computational complexity of the algorithm. Thanks to these changes, the learning converges on every run for different examples.

Training of the model is based on a few recent months of both real data and predictions of energy consumption, production from renewable energy sources and outside temperature. For calculating the reward to evaluate returned actions, a model of a home environment was used, with models of all end-devices simulating their work and the consumed energy. These were developed in previous cycles.

The AI algorithm consists now of two functions offloaded to the Serverless Runtime to run. Both were implemented in the Python language using the Torch package. The result of training is a Torch Module, which is supposed to be saved in DaaS. The model will be used in the decision-making function, which uses predictions for the future of energy signals and temperature to compute settings of appliances in a household network.

This version of the algorithm was deployed in the Python version of the demo. Simulations of different scenarios were tested and ran successfully using COGNIT Client.

Running the Device Client in C on the smart electricity meter

Since smart energy meters are devices with constrained resources, they can only use the Device Client in C. During this period, the Device Client and its dependencies were integrated into the Phoenix RTOS operating system. Moreover, the first simple function has been offloaded from the smart energy meter to the Serverless Runtime in the COGNIT Framework.

Significant effort was put by relevant partners into the development of the Device Client in C. We provided support for this development by testing a couple of versions of the Client on our devices and by giving comments and feedback about the solutions that are most suitable for our smart energy meters and potentially other similar devices. This involved e.g. the suggestion to use mbedtls instead of openSSL, since the former is much better suited for small, resource-constrained devices.

The integration of the Device Client into our devices focused mainly on the libraries that the Client and the example application depend on. This involved openSSL (in the early version of the Client), mbedtls and curl. All of these had already been ported into the Phoenix RTOS (see phoenix-rtos-ports on Github), but had not been used on our smart energy meter, nor on other devices with the same CPU architecture that the smart energy meter utilises.

Testbed setup

In our internal testbed located in a small village in Mazury (north-eastern part of Poland), some of the relevant devices were changed or upgraded. This involved:

- Additional photovoltaic panels.
- The energy storage was replaced by one with increased capacity.
- The inverters were replaced by newer versions of the devices.

The integration of our data collecting infrastructure with the new devices was successfully conducted.

Simulating multiple devices

The COGNIT Framework is required to be prepared for receiving requests from a vast number of devices in a short period of time. This agrees with the real-life scenario of our Use Case, which involves many smart energy meters all requesting function offloading in a small time frame. In order to simulate this situation, we have developed a script that launches a customizable number of instances of smart energy meter simulators, each of them sending requests to the COGNIT Framework with a predefined frequency. Additionally, a Docker image has been provided so that the use of the script is more straightforward and host-agnostic.

5.4 Integration with the COGNIT Framework

Application structure

Our scenario includes 2 types of functions offloaded to the COGNIT Framework:

- decision-making function,
- training of AI model.

They are both implemented in the Python language and require packages like numpy and torch for execution. The second library is especially important as it requires a lot of memory resources for installation. We needed to resize the disk connected to our flavour, increase RAM on the Virtual Machine and update image template.

The execution of the training function in the COGNIT Framework takes about 7-8 minutes, so increasing the time of nginx timeout was needed for it to work properly.

Integration with the COGNIT Framework

As described in previous sections, there has been significant progress in terms of the integration of the Device Client in C into the firmware of the smart energy meter. Offloading more complicated, use case-specific functions is an ongoing work.

Use Case Internal Testbed

Previous sections provide the status of the Use Case internal testbed in terms of the devices that are significant for the final demo. Setting up an internal Edge Cluster node is a work in progress.

Project Demo Use Case Testbed

The final demo involves offloading the AI-algorithm-specific functions from the smart energy meter. This requires some pending effort to be done in the matter of integrating the device with the COGNIT Framework. This is related to the following issues:

- The application running on the device, together with the Device Client in C, needs to handle offloading requests with functions that take a long time (e.g. 5-10 minutes) to run. This is essential when running the training function.
- The COGNIT Framework is required to provide a way to save and load the AI model to be used for training and inference when functions are offloaded from the device. Since smart energy meters are constrained in terms of RAM and flash memory, it would be hard to offload the whole AI model from the device application.
- The COGNIT Framework is required to store the user data and weather forecasts, and be able to download them from (external) **besmart.energy** service. This is used extensively by the AI algorithm training and inference.

6. Use Case #4: Cybersecurity

6.1 Implementation and Deployment Plan M22-M27

This is the Implementation and Deployment Plan made and executed by the Cybersecurity partners during the reporting period.

Implementation plan

- Configure and deploy an Edge Cluster of the COGNIT Framework (architecture v2) at CETIC with one local edge node at the testbed for UC4.
- Optimise Anomaly Detection (AD) model.
- Create a specific version of the use case implementation (containing the AD log model) (architecture v2).
- Integrate the AD log into the COGNIT Framework (Device Client + Serverless Runtime).
- Evaluate the scalability of the COGNIT Framework by simulating a large number of vehicles "on the road".

Detailed plan

- Rent hardware needed for local deployment of the testbed and test anomaly detection with a GPU.
- Configure a node in the Edge Cluster of the COGNIT Framework with a GPU to improve the performance of the anomaly detection model.
- Optimise the AD (Anomaly Detection) model by changing the approach. New model, more specialised, lighter and more adapted to an edged environment.
- Test the new architecture of the framework, by:
    - Using a simple anomaly detection function (inference).
    - Using an anomaly detection function calling the optimised model.
- Further improve the elements used by the Rover that are provided as parameters to the AD function, namely:
    - Generating authentication logs.
    - Retrieving authentication logs.
    - Offload and automatically execute the detection function for each new entry in the authentication logs.
- Running functions with new requirements, such as GPU to measure performance improvements.
- Deploying a 2nd edge node to prepare for the next cycle, during which we will demonstrate the framework's ability to manage mobility.

Validation criteria

- A COGNIT testbed (Edge Cluster) is deployed at CETIC and communicates with the RISE testbed, which is the full stack deployment of the framework at RISE servers.
- The Device Client is able to communicate with the framework (architecture v2) and to:
    - Retrieve new entries in the authentication logs.
    - Offload a simple anomaly detection function (inference).

    - Offload a complex anomaly detection function (AD model).
- The Serverless Runtime is able to:
    - Execute a simple anomaly detection function (inference).
    - Execute a complex anomaly detection function (AD model).
- The framework is able to consider new requirements and update the Serverless Runtime accordingly.

6.2 Use Case Scenario and Architecture

The use case scenario has been extensively described throughout previous deliverables (see D5.3 and D5.4).

Currently, the working assumptions regarding the constraints for the function execution concern mainly mobility, response time and scalability:

- The COGNIT Framework needs to consider the movement of vehicles and the functions performed for each of them. It is therefore necessary to predict the next node to use based on location and latency of the vehicle. It is expected that a list of alternative candidate Edge Clusters will be available from the COGNIT Frontend.
- Short response times will be expected for the anomaly detection functions running at the edge because anomalies need to be detected as early and as quickly as possible to allow time to react and recover before damage is caused by an attacker (e.g. accident or vehicle collision), i.e. <100ms for effective avoidance/mitigation and <50ms for dangerous conditions according to the ISO2626 automotive standard.
- Scalability because the number of vehicles is likely to change and each vehicle has 3 anomaly detection functions that need to be performed at the edge.

6.3 Summary of developments during cycle M22-M27

Observe changes (Data and Requirements) -> Trigger the offloading and the execution of the function

Two types of events influence the triggering of offloading and the execution of the anomaly detection function. On one hand, the update of the requirements, and on the other hand, the addition of a new block of lines in the log file by the rover system.

To monitor these changes, we developed and implemented two classes to watch and handle file modifications:

LogHandler:

- Monitors and reads changes in the log file (e.g., auth.log).
- Passes the detected events as parameters to the function.
- Triggers the offloading and execution of the function.

RequirementsHandler:

- Monitors changes in the requirements file (e.g., FLAVOUR, MAX_LATENCY, GEOLOCATION, etc.).

- Dynamically reloads the requirements and applies them so they are taken into account during the next triggering of offloading and function execution.

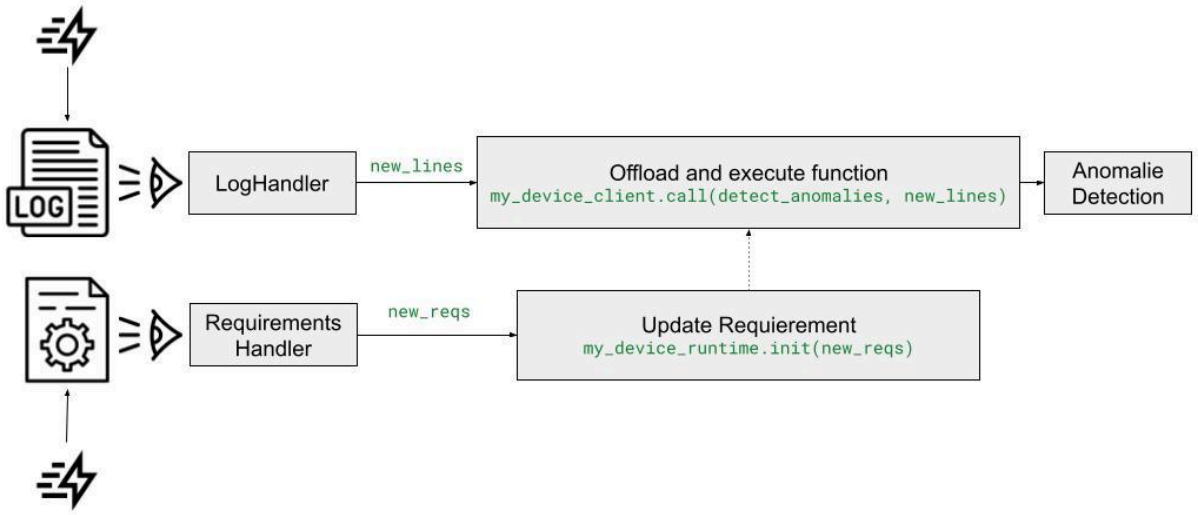


Figure 6.1. Observe changes (Data and Requirements) -> Trigger the offloading and the execution of the function

Detect anomalies (in Logs, GPS, Metrics data) with LLM

As soon as the client device detects a change in the logs, an anomaly detection request is offloaded to the COGNIT Framework. This anomaly detection relies on two complementary algorithms:

- A decision tree that provides a near real-time assessment of a potential anomaly and triggers remediation if necessary.
- An LLM that runs in parallel with the decision tree. Although slower, it performs a deeper analysis to identify new patterns, confirm or refute the anomaly, and, if needed, cancel any remediation previously initiated.

Give the appropriate response depending on the anomaly detection result

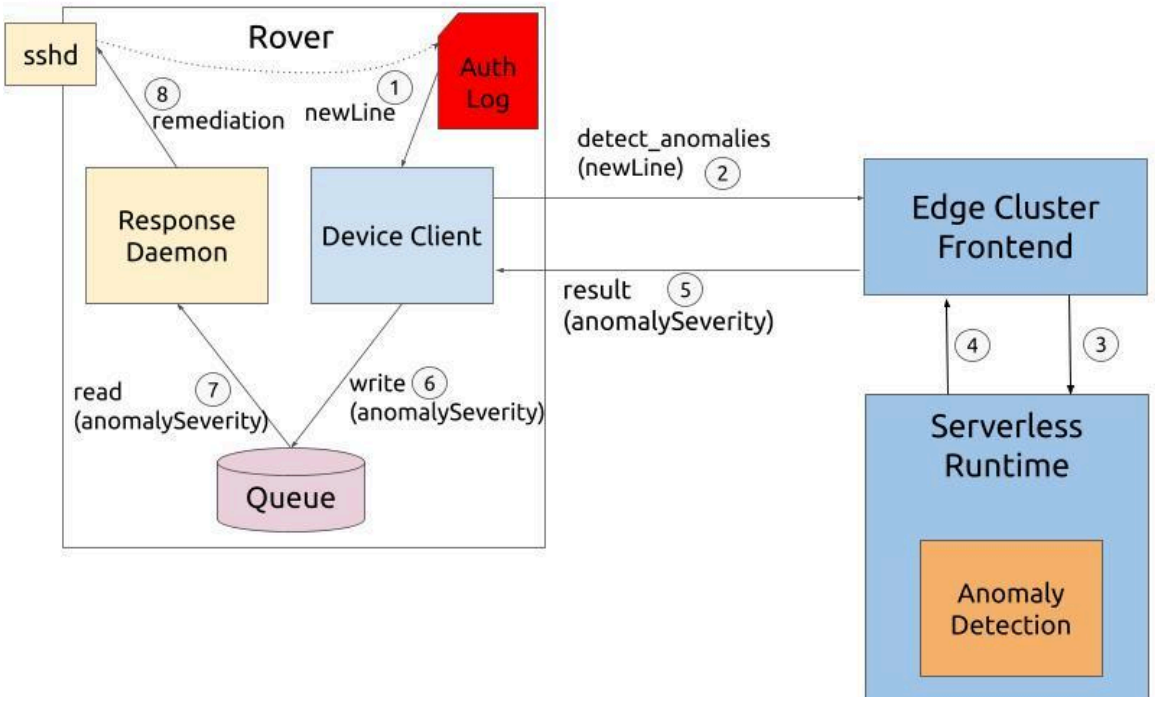


Figure 6.2. Integration and component interactions in UC4

The implementation consists of two primary components operating within the Rover environment:

1. **Device Client**: Monitors authentication logs, sends them to a serverless runtime for analysis, and processes the returned results.
2. **Response Daemon**: Consumes detection events and takes appropriate security actions.

The architecture also includes the Serverless Runtime environment that hosts the Anomaly Detection functionality.

Workflow

The system operates according to the following workflow:

1. When a new authentication log entry appears, it is detected by the Device Client.
2. The Device Client sends the new log line to the Edge Cluster Frontend.
3. The Edge Cluster Frontend forwards the request to the Anomaly Detection function in the Serverless Runtime.
4. After analysis, the Anomaly Detection function returns back the results.
5. The Edge Cluster Frontend returns the results to the Device Client, including severity levels for any detected anomalies.
6. If anomalies are detected, the Device Client writes appropriate events with severity information to a Queue.
7. The Response Daemon reads the Queue to obtain anomaly information and severity levels.
8. Based on the severity level, the Response Daemon implements remediation actions to the SSH daemon (sshd).

Response Actions

The Response Daemon implements two types of blocks based on severity levels:

- IP-specific blocks: Deny access to a user only from specific IP addresses (lower severity).
- Global blocks: Deny access to a user from all IP addresses (higher severity).

Blocks can be temporary (automatically expiring after a set duration) or permanent (requiring explicit unblock), depending on the severity level.

UC4 testing with Confidential Computing (Execute function in CC enclave)

In order to test and demonstrate the Confidential Computing (CC) support and capabilities of the COGNIT Framework, UC4 is making efforts to execute specific functions within a CC secured hardware enclave. UC4 is deploying a Confidential Computing-enabled testbed of the framework and plans to add a CC testing scenario into the demo in order to study it, assess compatibility, performance, and potential benefits. This scenario will have enabled the sensitive function execution requirements that will be passed to the COGNIT framework as part of a security strategy to enhance data protection and trust during sensitive processing tasks.

Federated learning consideration

To leverage the logs available on each client device, implementing a distributed training process (e.g., on a monthly basis) proves highly valuable. This approach enables the detection of new anomalies based on the real-world experience of the vehicles, while addressing the challenges of continual learning. Federated training would then be performed in a distributed environment at the edge to avoid exporting data from the client device to a central server.

6.4 Integration with the COGNIT Framework

Application structure

The overall structure of our application is detailed in deliverable D5.4, within section 6.3 (Integration with the COGNIT Framework), and the evolution of the application structure is outlined in the previous section (6.3 Summary of developments during cycle M22-M27).

To summarise, the main changes are focused on:

- Observing changes (Data and Requirements) and triggering offloading and function execution.
- Improving anomaly detection.
- Managing results (Appropriate Response).
- Considering the integration of Federated Learning for the models used in anomaly detection.

Furthermore, we will focus on the development of the simulator and the “digital twin” during the next cycle.

Integration with the COGNIT Framework

We created a new flavour CybersecV2, based on the first version, so that it is compatible with the new architecture (COGNIT V2).

Next, we modified this flavour to include the new dependencies specific to our new anomaly detection model.

We are now able to use the framework end-to-end, from triggering offloading to processing the results.

Use Case Internal Testbed

To validate the offloading process and the function execution using the new version of the COGNIT architecture and our flavour, CETIC maintains a local testbed environment in our premises for development purposes. This setup allows us to perform controlled tests and ensure compatibility with the updated architecture.

This environment runs on a VMWare vCenter environment composed of 10 DELL R450 servers. Each server is equipped with:

- CPU: Intel Xeon Gold 5317.
- RAM: 256 GB.
- Storage: A NAS system provides shared storage via two 20 TB iSCSI volumes.

On this infrastructure, we deployed three virtual machines to support our local testbed environment. Each VM has the following specifications:

- Frontend VM:
  - 2 vCPUs
  - 4 GB RAM
  - 60 GB disk
- Compute Node VM:
  - 8 vCPUs
  - 16 GB RAM
  - 250 GB disk
- Device Client VM:
  - 2 vCPUs
  - 4 GB RAM
  - 20 GB disk

The Device Client VM is used to simulate the rover and workloads. It hosts the response daemon and the log simulator. The COGNIT Device Client is deployed inside a Docker container on this VM, and includes the various components previously described (such as the file watchers and the COGNIT Device Client). All components run smoothly and communicate effectively within this environment.

The next planned step is the full deployment of the Edge Cluster and an additional edge node where Confidential Computing will be enabled. This will allow us to showcase the framework’s ability to demonstrate vehicle mobility across multiple nodes with different

security configurations and meet the other specific requirements of Use Case 4. The infrastructure acquisition (renting) process for this new deployment is currently underway.

Project Demo Use Case Testbed

The Cybersecurity pilot plans to leverage two complementary testbeds for its activities. First, it will utilise the testbed that has already been deployed at CETIC and integrated with the infrastructure at RISE ICE in Luleå, enabling the demonstration of data and process execution mobility as the rover moves between areas within network proximity of one testbed node to the other, impacting connection quality and latency.

In addition, the pilot will make use of the Confidential Computing testbed currently being deployed in a rented bare metal server in Belgium, which will allow the secure execution of sensitive functions within a trusted environment. Together, these testbeds will provide the environment for validating the cybersecurity use case in multiple node and CC enabled scenarios.

PART II. Software Integration and Verification

7. Software Integration Process and Infrastructure

OpsForge was updated to include the automation of the rabbitmq broker required by the rework made to the function offloading. The Edge Cluster Frontend Ansible role now includes the setup of this broker alongside the REST API. On top of this SSL termination with nginx has also been added as a default option to ease the deployment of each Edge Cluster Frontend.

7.1 COGNIT Frontend

The COGNIT Frontend can be deployed directly on bare metal or in an EC2 instance, depending on whether deploying on AWS or on-premises.

7.2 Edge Cluster Frontend

The Edge Cluster Frontend deployment can be done on any host. This automation is not part of the execution flow of the OpsForge tool, which only deploys the control plane. Since this component falls under the Edge Cluster (KVM nodes + Storage and Network)

scope, the deployment is done using dedicated Ansible playbooks, provided within the Ansible playbook stack in the OpsForge repository.

Ideally, it should be placed at a KVM node or a dedicated VM deployed in the Edge Cluster hypervisor nodes. The Edge Cluster Frontend needs to be reachable over the internet, so that the device client can request executions, and the broker needs to be reachable by the Serverless Runtimes VMs.

7.3 Third Version of the COGNIT Software Stack

The updated software components have been tagged with the appropriate version. The full deployment of v3.0 of the COGNIT stack can be performed using OpsForge from the tag release-cognit-3.0, using this same tag in all the components referred to in Table 7.1.

- https://github.com/SovereignEdgeEU-COGNIT/cognit-FrontEnd/releases/tag/release-cognit-3.0
- https://github.com/SovereignEdgeEU-COGNIT/edgecluster-FrontEnd/releases/tag/release-cognit-3.0

Name	Documentation	Testing	Installation
Device Client (Python)	Wiki documentation	Test folder	README
Device Client (C)	N/A	Test folder	README
COGNIT Frontend	User guide	Test folder	Install guide
Edge Cluster Frontend	User guide	Test folder	Install guide
Cloud-Edge Manager	Official doc	Q&A	Install guide
Serverless Runtime	Wiki documentation	Test folder	README
AI-Enabled Orchestrator	User guide	See docs	Install guide
COGNIT Opsforge	User guide	See docs	Install guide

Table 7.1. Main COGNIT 3.0 Framework components

8. Testbed Environment at RISE ICE Luleå

This section includes a brief summary of the developments of the main (COGNIT-LAB) testbed in RISE ICE, Luleå, during the M22-M27 cycle to support the new version (v2) of the COGNIT architecture.

The full testbed status and configuration at the M27 milestone is documented in an internal GitHub repository.

8.1 COGNIT Frontend

This new component is hosted in the existing VM that was previously hosting the Provisioning Engine (a deprecated component from v1 of the COGNIT architecture). Another public API endpoint was added, to make the component accessible.

8.2 Edge Cluster Frontend

One VM was created in the ICE Edge Cluster to host the new Edge Cluster Frontend, which is a new mandatory component for all Edge Clusters. A public IP was allocated and assigned to this VM to expose it to the public Internet.

8.3 Public DaaS

S3 is used for data sharing across the cloud-edge continuum. MinIO (https://min.io) is used to implement COGNIT global DaaS. To support the public MinIO requirement, a deployment was created at RISE ICE in Luleå which is reachable via a public endpoint.

The current MinIO deployment is single-node. An additional VM was added for this purpose with the following initial size:

- 4 CPU
- 16 GB RAM
- 1x100GB XFS volume for MinIO data

The plan is to scale up the VM on demand and if that is not enough, then the deployment can be converted to a horizontally scalable multi-node production deployment.

8.4 Edge private DaaS

Every Edge Cluster needs to host one local MinIO instance as private DaaS. For ICE Edge Cluster a new VM was created within the ICE Edge Cluster. MinIO was installed and made accessible by local clients.

9. Software Requirements Verification

Possible status are: NOT STARTED | IN PROGRESS | COMPLETED

9.1 Device Client

SR1.1 Interface with COGNIT Frontend
Status: IN PROGRESS
Description: Implementation of the communication of the Device Client with the COGNIT Frontend. Following the instructions[4] of the Project's GitHub repository, a user can authenticate, update application requirements and get an Edge Cluster Frontend IP address to offload the execution of a Python function. All the library functionalities can be tested standalone by executing the unit tests provided on the GitHub repository.[5]

Completed Verification Scenarios:

- [VS1.1.1] The Device Client is able to get authorisation from COGNIT and is able to send App valid requirements to the COGNIT Frontend.
- [VS1.1.2] The Device Client is able to receive a valid Edge Cluster (effectively a valid Edge Cluster Frontend IP address).
- [VS1.1.3] The Device Client is able to update the App requirements at any moment.
- [VS1.1.4] The Device Client is able to receive a changed Edge Cluster seamlessly from a COGNIT's proactive decision-making action.

Pending Verification Scenarios:

- [VS1.1.5] The Device Client is able to handle (upload/read) data on the COGNIT global layer.

SR1.2 Interface with Edge Cluster
Status: IN PROGRESS
Description: Implementation of the communication of the Device Client with the Edge Cluster.

[4] https://github.com/SovereignEdgeEU-COGNIT/device-runtime-py/blob/main/README.md
[5] https://github.com/SovereignEdgeEU-COGNIT/device-runtime-py/tree/main/cognit/test

Completed Verification Scenarios:

- [VS1.2.1] The Device Client is able to execute functions (either preloaded or uploading it at the moment of execution) on the assigned Edge Cluster.

Pending Verification Scenarios:

- [VS1.2.2] The Device Client is able to handle (upload/read) data privately in the assigned Edge Cluster.

SR1.3 Programming languages

Status: IN PROGRESS

Description: Support for different programming languages.

Completed Verification Scenarios:

- VS[1.3.2] Test VS1.1.1 to VS1.1.4 and VS1.2.1 validation scenarios both in C and Python version of the Device Client.

Pending Verification Scenarios:

- VS[1.3.1] Test VS1.1.5 and VS1.2.2 validation scenarios both in C and Python version of the Device Client.

SR1.4 Low memory footprint for constrained devices

Status: IN PROGRESS

Description: Low memory footprint for constrained devices.

Completed Verification Scenarios:

- VS[1.3.2] Test VS1.1.1 to VS1.1.4 and VS1.2.1 validation scenarios in the C version of the Device Client.

Pending Verification Scenarios:

- [VS1.4.1] Test VS1.1.5 and VS1.2.2 validation scenarios on a device with less than 500kB of RAM, making use of the Device Client in C.

SR1.5 Security

Status: IN PROGRESS

Completed Verification Scenarios:

- [VS1.5.1] The Device Client is able to perform secure communications against the COGNIT Frontend and the assigned Edge Cluster Frontend with the acceptance of the authorization mechanism.

Pending Verification Scenarios:

- [VS1.5.2] The Device Client is not permitted any unauthorised action towards the COGNIT Frontend or the assigned Edge Cluster.

SR1.6 Collecting Latency Measurements

Status: NOT STARTED

Pending Verification Scenarios:

- [VS1.6.1] The Device Client is able to measure latencies to different Edge Clusters concurrently with other activities of the Client, to be able to monitor and make COGNIT aware of the effective latency of the potential Edge Clusters to be used.

9.2 COGNIT Frontend

SR2.1 COGNIT Frontend

Status: IN PROGRESS

Description: Provides an entry point for devices to communicate with the COGNIT Framework for offloading the execution of functions and uploading global data.

Completed Verification Scenarios:

- [VS2.1.1] Authenticate a Device against the COGNIT Frontend and verify that an authorization token is returned.
- [VS2.1.2] Upload application requirements to the COGNIT Frontend and verify that a unique ID is returned for the application requirements.
- [VS2.1.3] Upload application requirements and query the COGNIT Frontend for an Edge Cluster and verify that it meets the application requirements.

- [VS2.1.4] Upload a function to the COGNIT Frontend and verify that a unique ID is returned for that function.

Pending Verification Scenarios:

- [VS2.1.5] Test uploading and downloading data by the device to and from the COGNIT Frontend.

9.3 Edge Cluster

SR3.1 Edge Cluster Frontend

Status: IN PROGRESS

Description: The Edge Cluster must provide an interface (Edge Cluster Frontend) for the Device Client to offload the execution of functions and to upload local data that is needed to execute the function.

Completed Verification Scenarios:

- [VS3.1.1] Instantiate a Serverless Runtime and verify that a device can request the execution of a function to the Edge Cluster Frontend and assert the result of the function.

Pending Verification Scenarios:

- [VS3.1.2] Test uploading and downloading data by the device to and from the Edge Cluster using a secure communication channel.

SR3.2 Secure and Trusted Serverless Runtimes

Status: COMPLETED

Description: The Serverless Runtime is the minimal execution unit for the execution of functions offloaded by Device Clients.

Completed Verification Scenarios:

- [VS3.2.1] Build a Serverless Runtime image, customised for each Use Case, in an automated way.

9.4 Cloud-Edge Manager

SR4.1 Provider Catalog

Status: NOT STARTED

Description: Implement a backend to persist information about the available providers that can be used to extend the capacity of the COGNIT infrastructure.

Pending Verification Scenarios:

- [VS4.1.1] Listing the providers belonging to the Provider Catalog.
- [VS4.1.2] Filtering the providers according to a desired latency threshold on a geographic area.
- [VS4.1.3] Filtering the providers according to a cost per hour threshold.
- [VS4.1.4] Filtering the providers according to energy consumption per hour threshold.
- [VS4.1.5] Filtering the providers according to some specific hardware characteristics (e.g. GPUs, Trusted Execution Environments).

SR4.2 Edge Cluster Provisioning

Status: NOT STARTED

Description: The Cloud-Edge Manager must be able to provision Edge Clusters as a set of software-defined compute, network, storage on any cloud/edge location available in the Provider Catalogue.

Pending Verification Scenarios:

- [VS4.2.1] A YAML file containing the information about the provision is provided to the Cloud-Edge Manager that creates a new Edge Cluster.
- [VS4.2.2] Query the Cloud-Edge Manager to return the status of an Edge Cluster identified by its ID.
- [VS4.2.3] Query the Cloud-Edge Manager to scale up/down the number of hosts of an Edge Cluster identified by its ID.
- [VS4.2.4] Query the Cloud-Edge Manager to delete an Edge Cluster identified by its ID.

SR4.3 Serverless Runtime Deployment

Status: COMPLETED

Description: The Cloud-Edge Manager must be able to deploy Serverless Runtimes as Virtualized Workloads within an Edge Cluster.

Completed Verification Scenarios:

- [VS4.3.1] A YAML file containing the information about the deployment is provided to the Cloud-Edge Manager that creates a new Serverless Runtime.
- [VS4.3.2] Query the Cloud-Edge Manager to return the status of a Serverless Runtime identified by its ID.
- [VS4.3.3] Query the Cloud-Edge Manager to scale up/down the resources (CPU, memory and disks) of a Serverless Runtime identified by its ID.
- [VS4.3.4] Query the Cloud-Edge Manager to update the deployment of the Serverless Runtime identified by its ID.
- [VS4.3.5] Query the Cloud-Edge Manager to delete a Serverless Runtime identified by its ID.

SR4.4 Metrics, Monitoring, Auditing

Status: IN PROGRESS

Description: Edge-Clusters monitoring, Serverless Runtimes metrics collection and continuous security assessment.

Completed Verification Scenarios:

- [VS4.4.1] Create an Edge Cluster and deploy a Serverless Runtime and check the metrics collected for a certain period of time.

Pending Verification Scenarios:

- [VS4.4.1] Correct measuring of network latency from the Device Client to a particular Serverless Runtime

SR4.5 Authentication & Authorization

Status: COMPLETED

Description: Authentication and authorization mechanisms for accessing cloud-edge infrastructure resources by the devices for offloading workloads.

Completed Verification Scenarios:

- [VS4.5.1] Test the creation of new users and groups
- [VS4.5.3] Communicate with the COGNIT Frontend and the Edge Cluster Frontend using tokens.
- [VS4.5.2] Assign ACLs to designated users and test the creation of new Edge Clusters and Serverless Runtimes.

SR4.6 Plan Executor

Status: IN PROGRESS

Description: The Plan Executor is responsible for converting plans provided by the AI-Enabled Orchestrator in Cloud-Edge Manager actions for the life cycle management of Edge Clusters and Serverless Runtimes.

Completed Verification Scenarios:

- [VS4.6.1] Submit a plan to the Plan Executor for creating new Serverless Runtimes and verify the deployment of the Serverless Runtimes.
- [VS4.6.3] Submit a plan to the Plan Executor for resizing and migrating a Serverless Runtime.

Pending Verification Scenarios:

- [VS4.6.2] Submit a plan to the Plan Executor for creating a new Edge Cluster and verify that the Edge Cluster is created correctly.
- [VS4.6.4] Submit a plan to the Plan Executor for increasing the number of hosts (horizontal scaling) of an existing Edge Cluster.

9.5 AI-Enabled Orchestrator

SR5.1 Building Learning Models

Status: COMPLETED

Description: Provide AI/ML models trained with input from collected metrics from the

Cloud-Edge Manager monitoring service related to Edge Clusters and Serverless Runtimes deployed across the distributed cloud-edge continuum.

Completed Verification Scenarios:

- [VS5.1.1] List instances from Devices to Applications to System for metrics to be collected.
- [VS5.1.2] Correlate and represent features that are ready to take as input to the Model.
- [V1S5.1.3] Feedback-aware performance check when training the model on represented features.
- [VS5.1.4] Assess the ability in terms of AUROC score for each task (e.g. scheduling).

SR5.2 Smart Management of Cloud-Edge Resources

Status: IN PROGRESS

Description: The AI-Enabled Orchestrator is responsible for the automated management of cloud-edge continuum resources in order to optimize the performance of the applications that are offloading functions to the COGNIT Framework.

Pending Verification Scenarios:

- [VS5.2.1] Assess the ability of workload and resource optimization in terms of cost and performance trade-off.

9.6 Secure and Trusted Execution of Computing Environments

SR6.1 Advanced Access Control

Status: IN PROGRESS

Description: Implement policy-based access control to support security policies on geographic zones that define a security level for specific areas.

Pending Verification Scenarios:

- [VS6.1.1] Define a security policy that is based on geographic zone attributes.
- [VS6.1.2] Check enforcement of new security policy when edge device moves

closer from one edge node than another.

SR6.2 Confidential Computing

Status: IN PROGRESS

Description: Enable privacy protection for the application workloads at the hardware level using Confidential Computing (CC) techniques.

Pending Verification Scenarios:

- [VS6.2.1] Deploy a function on a host that provides confidential computing capability.
- [VS 6.2.2] Check that the function is executed inside the host trusted execution environment (TEE).

SR6.3 Federated Learning

Status: NOT STARTED

Description: Enhance privacy of AI workloads that have confidentiality requirements preventing the exchange of information for training. Federated Learning techniques enable confidential or private data processing under the control of the data owner or controller, with only learned models shared.

Pending Verification Scenarios:

- [VS6.3.1] Perform training of the ML algorithm without exchanging local data.
- [VS6.3.2] Check that the redistributed models for inference do not contain private data.

10. Conclusions and Next Steps

During the reporting period, the project Use Cases have made significant progress in adapting their applications to the updated COGNIT project architecture. New functionality for running COGNIT on hardware-constrained devices has been made available with function offloading capabilities from clients running in C. The process of testing their applications in both simulations and with the target hardware devices has started. Testing has been performed at the shared project testbed at ICE, and the process of setting up and testing the local edge nodes has been started.

The next steps for the Use Cases are continued testing and development toward final project demonstrators. This includes participating in scalability testing of the project infrastructure and the evaluation of the AI-enabled orchestrator.