Applications Performance Analysis in C-ITS

PETRU LUCIAN SANDU
(Licenciado em Engenharia Eletrónica e Telecomunicações e de Computadores)

Dissertação para obtenção do grau de Mestre em Engenharia Eletrónica e Telecomunicações, no Perfil de Telecomunicações

Orientadores:

Doutor José Manuel de Campos Lages Garcia Simão
Doutor António João Nunes Serrador

Júri:

Presidente: Doutor Rui António Policarpo Duarte
Vogais:
Doutor António João Nunes Serrador
Doutor João Miguel Pereira de Almeida

Outubro 2025

(page intentionally left blank)

Applications Performance Analysis in C-ITS

PETRU LUCIAN SANDU
(Licenciado em Engenharia Eletrónica e Telecomunicações e de Computadores)

Dissertação para obtenção do grau de Mestre em Engenharia Eletrónica e Telecomunicações, no Perfil de Telecomunicações

Orientadores:

Doutor José Manuel de Campos Lages Garcia Simão, Instituto Superior de Engenharia de Lisboa
Doutor António João Nunes Serrador, Instituto Superior de Engenharia de Lisboa

Júri:

Presidente: Doutor Rui António Policarpo Duarte, Instituto Superior de Engenharia de Lisboa
Vogais:
Doutor António João Nunes Serrador, Instituto Superior de Engenharia de Lisboa
Doutor João Miguel Pereira de Almeida, Universidade de Aveiro

Outubro 2025

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisors, ***Professor José Simão*** *and* ***Professor António Serrador****, for their guidance and support throughout the development of this thesis. Their constant availability to discuss and solve any issues, as well as their close involvement in every stage of the work, were fundamental to the successful completion of this project.*

I would also like to extend a special thanks to ***João Silva*** *and* ***Nuno Lopes*** *from* ***Infraestruturas de Portugal****, for their readiness and dedication in accompanying the testing process, always swift and efficient in resolving any unexpected problems, even while carrying out their many other professional responsibilities. I am also grateful to* ***Infraestruturas de Portugal*** *itself for providing the facilities, equipment, and systems that made the practical implementation of this work possible.*

Finally, but by no means less important, I would like to thank my ***girlfriend*** *and my* ***family*** *for their unconditional support and patience throughout this journey, especially during its most demanding and challenging moments.*

STATEMENT OF INTEGRITY

I declare that this dissertation report is the result of my personal and independent research. Its content is original, and all sources listed in the bibliographic references were consulted and are duly mentioned in the text. I further declare that all scientific and technical references relevant to the development of the work are duly cited and included in the bibliographic references.

The author

Assinado por: **Petru Lucian Sandu**
Num. de Identificação: CR:PT-1S5S49561
Data: 2025.10.22 12:12:02+01'00'

Lisbon, October, 2025

Resumo

O rápido avanço da tecnologia de transporte permitiu a implementação dos Sistemas de Transporte Inteligentes Cooperativos (C-ITS), melhorando a segurança rodoviária, o fluxo de tráfego e a eficiência global. No entanto, as implementações atuais são concebidas para veículos com Unidades de Bordo (OBUs) dedicadas, o que limita o acesso dos veículos mais antigos. Esta limitação constitui um obstáculo significativo à adoção generalizada e à plena concretização do potencial dos C-ITS. Esta dissertação avalia a arquitetura C-ITS atualmente em desenvolvimento pela Infraestruturas de Portugal (IP), com foco no seu desempenho, escalabilidade e capacidade de suportar uma adoção mais abrangente.

Foi desenvolvida uma aplicação móvel para simular a experiência do utilizador final, subscrevendo tópicos C-ITS através do protocolo MQTT, permitindo assim uma avaliação prática da entrega de mensagens a dispositivos para além das OBUs dedicadas. Foram realizados testes de desempenho abrangentes, incluindo análise de latência, testes de carga e avaliação da resiliência perante falhas de componentes.

Esta investigação contribui para a documentação e avaliação de uma implementação real de C-ITS, destacando as principais áreas de otimização. Em última análise, apoia os esforços em curso para expandir os C-ITS e garantir um acesso mais inclusivo à informação de tráfego para todos os veículos.

Palavras-chave: C-ITS, Veículos Legados, MQTT, ITS-G5, Cellular V2X, Unidades de Beira de Estrada (RSU), Transporte Inteligente, Segurança, Comunicação em Tempo Real.

Abstract

The rapid advancement of transport technology has enabled the deployment of Cooperative Intelligent Transport Systems (C-ITS), enhancing road safety, traffic flow, and overall efficiency. However, current implementations are designed for vehicles with dedicated On-Board Units (OBUs), limiting access for legacy vehicles. This presents a major barrier to widespread adoption and the full potential of C-ITS. This thesis evaluates the C-ITS architecture currently under development by Infraestruturas de Portugal (IP), focusing on its performance, scalability, and ability to support broader adoption.

A mobile application was implemented to simulate the end-user experience, subscribing to C-ITS topics through MQTT and enabling practical assessment of message delivery to devices beyond dedicated OBUs. Comprehensive performance tests were conducted, including latency analysis, load testing, and resilience under component failures.

This research contributes to the documentation and evaluation of a real-world C-ITS deployment, highlighting key areas for optimisation. Ultimately, it supports the ongoing efforts to scale C-ITS and ensure more inclusive access to traffic information for all vehicles.

Key Words: C-ITS, Legacy Vehicles, MQTT, ITS-G5, Cellular V2X, Roadside Units (RSU), Intelligent Transportation, Security, Real-time Communication.

Table of Contents

List of Figures

List of Tables

1 Introduction

The continuous evolution of urban areas and the increasing number of vehicles on the road have amplified the need to improve road safety. While the automobile industry has been making efforts to implement advanced safety technologies, such as lane-keeping systems, these solutions alone are not sufficient. Therefore, the importance of constant communication through a reliable infrastructure that provides real-time information about potential dangers or imminent situations has become more evident. Beyond safety concerns, the rising costs of vehicle ownership, from fuel consumption to the replacement of wearable components, further highlight the need for efficiency. By leveraging the C-ITS network, vehicles can access real-time information to optimize driving behavior, such as reducing unnecessary acceleration. This conserves fuel and minimizes mechanical wear, ultimately lowering maintenance costs.

These advancements in real-time vehicle communication also lay the foundation for the future of autonomous driving. As C-ITS technology evolves, it enhances collaboration between vehicles and infrastructure, further improving safety, efficiency, and overall traffic management.

1.1 Motivation

According to the European Commission statistics, in 2023, 20 380 persons were killed in road accidents in the EU, an increase of 3.7% compared with 2021 [1]. Additionally, for every road fatality, it is estimated that there are four permanently disabling injuries, such as brain or spinal cord damage. The implementation of Intelligent Transportation Systems (ITS), which assist drivers and enable vehicle connectivity, has the potential to significantly reduce accidents and improve overall road safety [2]. Also, in the current implementation of the C-ITS network, only newly released vehicles equipped with a compatible device can connect to the system. Vehicles that do not have a pre-installed OBU, which we will refer to in this thesis as ”Legacy Cars”, are unable to access the network, making C-ITS an exclusive functionality limited to certain vehicles. However, this exclusivity contradicts the main purpose of the technology, highlighting the need for a hybrid solution that enables both Legacy Cars and pre-connected vehicles to use the C-ITS network.

This thesis explores such a solution, enhancing scalability and ensuring broader

adoption of this technology. Furthermore, allowing Legacy Cars to receive reliable roadway information, improving overall traffic safety and efficiency.

1.2 Objectives

The primary objective of this research is to evaluate the C-ITS architecture proposed by *Infraestruturas de Portugal*(IP), focusing on its latency performance and system bottlenecks. This study aims to identify weak points within the architecture and propose potential optimizations to enhance real-time message delivery.

A key aspect of this research involves simulating the experience of a final user, by developing and integrating a mobile application capable of subscribing to the correct C-ITS topics and consuming messages efficiently.

To achieve this, the study will:

- Measure and analyze latency within the IP architecture to determine critical delays and identify performance constraints.
- Develop a mobile application that acts as a C-ITS consumer, subscribing to relevant MQTT topics and processing real-time data.
- Evaluate message delivery speed and assess how efficiently data is transmitted using ITS-G5 technology
- Propose optimizations to mitigate bottlenecks and improve system efficiency, ensuring a seamless experience for final users.

By combining technical performance analysis with real-world user interaction, this research will contribute to developing a scalable and efficient C-ITS solution that can effectively integrate with existing infrastructure while improving message delivery and usability. Overall it will ensure that the system functions as expected for end users while maintaining low latency and reliable communication.

1.3 Planning

The development of this thesis will follow the Gant chart presented in Figure 1.1. There may be some deviations in the exact course of work development, but overall, it is expected to follow this structure.

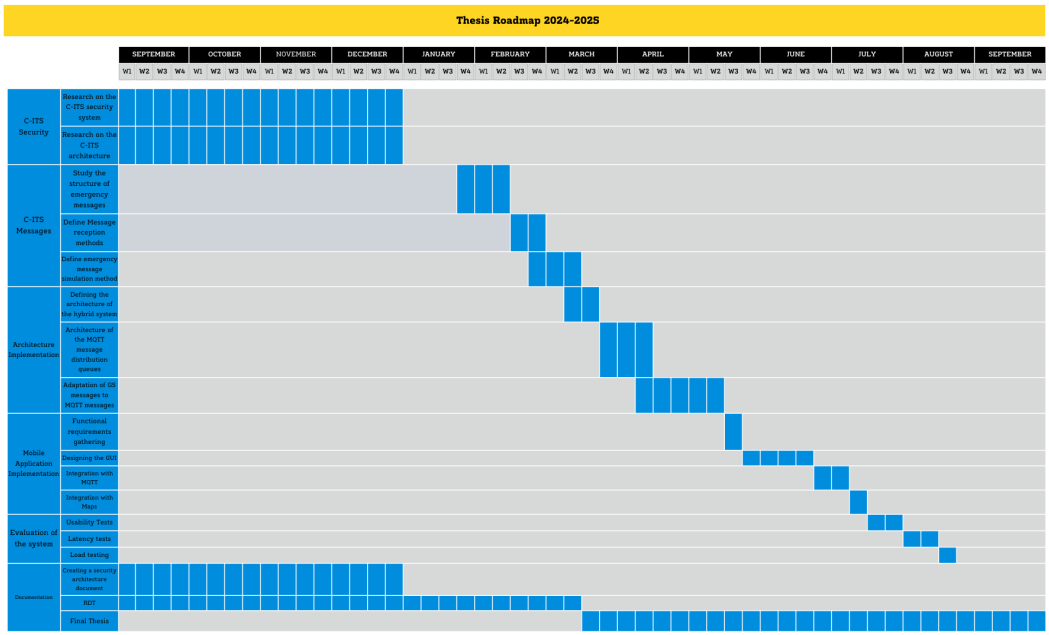


Figure 1.1: Public Key Infrastructure Architecture

1.4 Contributions

This research project has already contributed to the academic and professional community by preparing and submitting an abstract for the First International Conference on Transportation Systems (TS2025). The abstract was approved and the project was presented at the conference, ensuring international visibility.

Beyond academic contributions, this project also addresses an industry need. *Infraestruturas de Portugal* requires documentation on their C-ITS architecture, which is currently under development. As part of this research, we contributed to refining the documentation that supports their system's deployment and operational aspects, while also assisting IP in carrying out essential tests that were fundamental for the improvement and validation of their system.

2
State Of the Art

This chapter provides an overview of the current state of C-ITS, beginning with its historical context and key European initiatives that have driven its deployment. The chapter then explores the fundamental components of C-ITS, including system architecture, service levels, security mechanisms, and message types. Additionally, it highlights related research efforts in hybrid communication approaches, particularly the integration of C-ITS with message-oriented solutions like MQTT. Lastly, the main technical challenges faced in this project are presented, covering aspects such as IP architecture, MQTT implementation, and mobile application development.

2.1 Historical Context

One of the most well-known projects in the C-ITS domain is C-Roads, launched in 2016 and funded by the EU Connecting Europe Facility (CEF) program. This initiative brings together European Member States and road operators to deploy interoperable C-ITS services across Europe by standardizing infrastructure elements and defining a common service portfolio, thereby accelerating the deployment [3]. The evolution of C-ITS has followed a gradual approach, beginning with ”Day-1 services,” which encompass simpler but essential use cases. These services include messages for traffic congestion alerts, hazard warnings, roadworks notifications, slow or stationary vehicle reports, weather conditions, and speed recommendations to improve traffic harmonization. The information is delivered in a way that ensures drivers are informed without being distracted [4]. Although by the conclusion of the C-Roads project, C-ITS was still in the deployment phase rather than being fully operational across Europe, significant progress had been made [5]. Similar developments were also observed globally, particularly in Asia [6].

Recognizing the need for further expansion, Portugal launched the C-Streets project as a successor to C-Roads, focusing on testing and implementing C-ITS services in urban and metropolitan environments. While C-Roads primarily addressed highways, C-Streets aimed to bridge the gap by integrating C-ITS solutions into urban mobility ecosystems, ensuring service continuity across different transport infrastructures [7].

While C-ITS has made significant progress in deployment, a key challenge re-

mains in ensuring that its information reaches a broader audience beyond equipped vehicles. Current deployments rely heavily on ITS-G5 and cellular technologies for direct vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication. However, these solutions often exclude legacy vehicles and other road users who lack built-in C-ITS connectivity. To address this limitation, new approaches—such as hybrid architectures integrating C-ITS with complementary technologies—are emerging to bridge this gap and extend the system's reach.

2.2 The C-ITS System

C-ITS refers to transportation systems in which two or more Intelligent Transport System (ITS) sub-systems, such as personal, vehicle, roadside, and central systems, work together. This cooperation enhances the quality and service level of ITS services compared to those provided by a single sub-system [8].

2.2.1 C-ITS System Architecture

Several C-ITS reference architectures have been proposed in recent years to support large-scale deployment across regions and nations. This architecture, provides a generalized model for C-ITS infrastructure, primarily addressing traffic-related concerns, and categorizes C-ITS systems into five main components [9] that can be seen on Figure 2.1:

1. Support System - Manages C-ITS services' governance, security, and operational aspects.

2. Central System - Captures and processes data from vehicles and roadside infrastructure.

3. Roadside System - Includes physical infrastructure like RSUs, traffic controllers, and surveillance cameras.

4. Vehicle System - Encompasses onboard units (V-OBUs) installed in vehicles.

5. Traveler/VRU System - Involves personal devices such as smartphones used by vulnerable road users (VRUs).

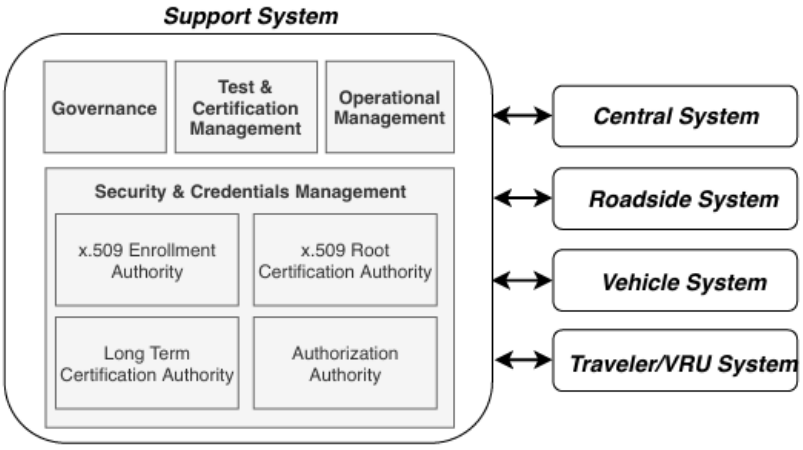


Figure 2.1: C-ITS Reference Architecture (extracted from [9])

The Support System is crucial in security and credential management, ensuring trusted communication and data protection across C-ITS components. The security and credential management subsystem will be discussed further in 2.2.3.

2.2.2 Service Levels of TLM

This chapter provides a detailed explanation of the 3 levels of implementation of the Trust List Manager (TLM), highlighting their main operational functionalities and their role within the overall architecture. The information is based on [10] and [11].

- **Level 0:** This level is intended primarily for short-term testing (shorter than 2 months) and interoperability sessions. It does not require an audit of the Certificate Policy (CP), meaning devices can subscribe to this level without needing to fully comply with standard security and operational protocols. This level allows flexibility for rapid testing and early-stage interoperability checks without imposing strict security requirements.

- **Level 1:** This level is intended for longer operational periods, allowing for certificates with extended validity (typically 3-6 months or more). While no CP audit is required at this level, the requirements and processes are closely aligned with full CP compliance, with only a few well-defined exceptions permitted as specified by the Certificate Policy Audit (CPA). This makes Level 1 more flexible than full compliance (Level 2), while still maintaining a high standard of security. It is suitable for early-stage real-world applications that require moderate security and reliability but do not yet need the full rigor of Level 2.

- **Level 2:** This level is enabled by the CPA and operates in full accordance with the CP, with no exceptions. A comprehensive audit is required, meaning entities at this level must undergo thorough evaluations to ensure they meet all CP standards without exceptions. This level is necessary for full-scale,

high-security operations that demand the highest degree of compliance and reliability, suitable for large-scale and mission-critical C-ITS deployments.

2.2.3 Security in C-ITS

Security is a fundamental aspect of C-ITS, ensuring trust, authentication, and confidentiality between participating entities. The Public Key Infrastructure (PKI) architecture plays a central role in this ecosystem. This chapter provides an overview of the PKI framework, illustrated in Figure 2.2, and its key components. The information presented is based on [12].

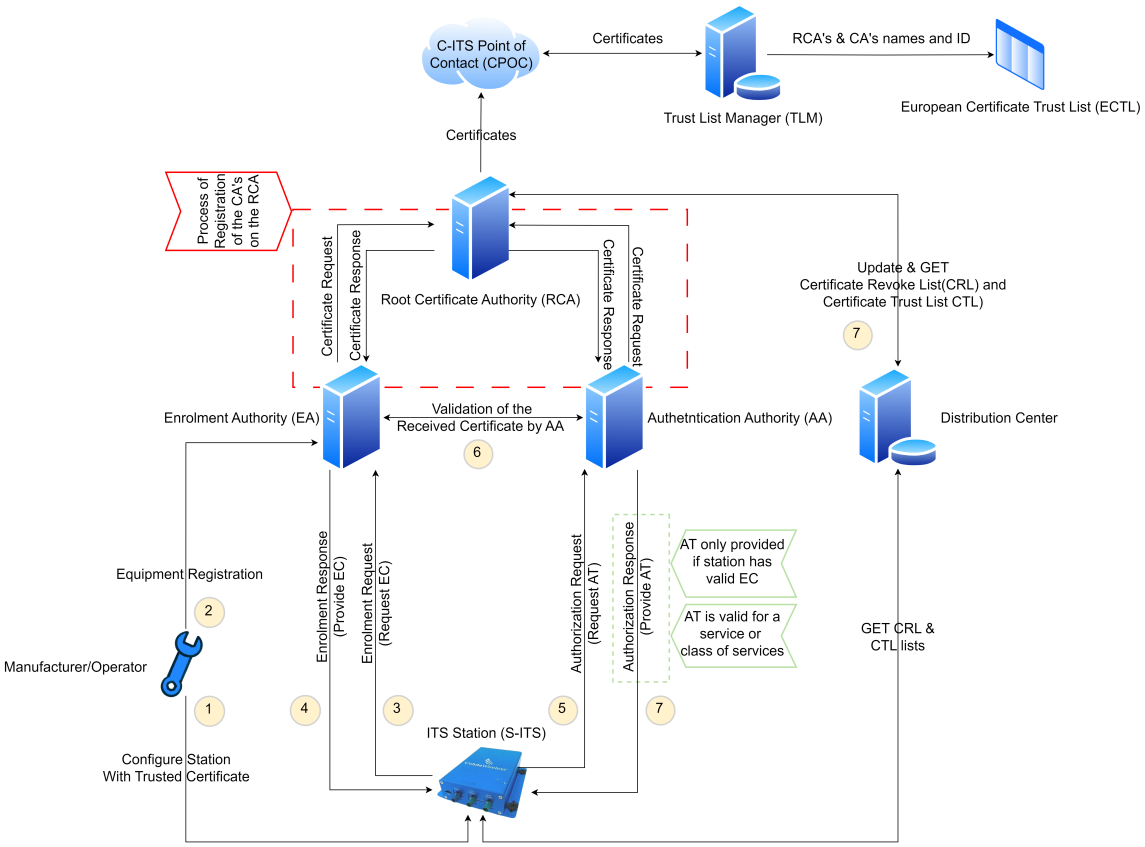


Figure 2.2: Public Key Infrastructure Architecture (based on [12])

Trust List Manager

Starting from the top we have the TLM. This is the entity responsible for issuing and managing the European Certificate Trust List (ECTL). The ECTL contains all the trusted Certificate Authorities (CAs) in Europe, including Root Certificate Authorities (RCAs), Enrolment Authorities (EAs), and Authorization Authorities (AAs), along with the public certificates of each authority. This list enables the verification of certificates issued by these authorities, ensuring that only trusted entities participate in the C-ITS ecosystem.

C-ITS Point of Contact

The information about the trusted entities comes from the C-ITS Point of Contact (CPOC) which is responsible for "establishing and contributing to the secure communication exchange between all entities of the C-ITS trust model, reviewing the procedural change requests and recommendations submitted by other trust model participants and transmitting the Root-CA certificates to the Trust List Manager" [11].

Root Certificate Authority

The Root Certificate Authority (RCA) is responsible for registering subordinate Certificate Authorities, such as Enrolment Authorities (EAs) and AAs, in the Certificate Trust List (CTL) and Certificate Revocation List (CRL). By being included in these lists, these subordinate CAs are recognized as trusted entities authorized to issue certificates within the C-ITS ecosystem. The RCA also publishes this trust (CTL) and revocation (CRL) lists to the Distribution Center (DC), which provides access to the CTL and CRL for all devices in the ITS system. This ensures that all entities within the C-ITS network can verify trusted authorities and revoke untrusted ones, maintaining a secure and reliable environment.

Authorization Authority

The AA issues Authorization Tickets (ATs) to C-ITS stations based on the trust established during the enrolment process by the Enrolment Authority (EA). These tickets serve as proof that a C-ITS station is authorized to access and use specific ITS services. The digital signatures created with these tickets ensure message authenticity and provide non-repudiation. The signing process uses cryptographic techniques embedded in the authorization tickets to create secure, verifiable digital signatures for each transmitted message.

Enrolment Authority

The EA (Enrolment Authority) is responsible for verifying and enrolling new devices in the network and generating Enrolment Certificates (EC) for ITS stations. These are long-term certificates that contain the station's real identity and are required to request an Authorization Ticket from the AA. EAs are also responsible for the revocation of the EC and managing the permissions of the ITS-S.

Manufacturer/Operator

The Manufacturer/Operator (Equipment Manufacturers, ITS Station Vendors, Integration Companies) is responsible for the initial configuration of ITS stations, embedding critical cryptographic elements and identity information during production. This includes [13]:

- Assigning a canonical identifier.

- Contact information for the EA and AA which will issue certificates for the ITS-S (network address and public key certificate).

- The set of currently known trusted AA certificates that the ITS-S may use to trust communications from other ITS-S.

- A public/private key pair for cryptographic purposes (canonical key pair).

- The trust anchor (Root CA) public key certificate and the DC network address.

- In case of a multiple root CAs architecture, the TLM public key certificate and the CPOC network address.

The manufacturer registers the station's public key and profile information with the Enrolment Authority (EA) to authorize its participation in the C-ITS ecosystem. This process ensures that the station is securely integrated into the system and can establish trusted communication upon deployment. The registration includes the following key elements [13]:

- The permanent canonical identifier of the ITS-S.

- The profile information for the ITS-S may contain an initial list of maximum appPermissions (permissions of the station), region restrictions, and assurance level which may be modified over time.

- The public key from the key pair belonging to the ITS-S (canonical public key).

2.2.4 Types of messages in C-ITS

CAM Message

The following CAM information is based on [14]. Cooperative Awareness Messages (CAMs) are messages transmitted within the ITS network between ITS stations (ITS-S). The main objectives of a CAM message are to establish and maintain mutual awareness and facilitate cooperative vehicle operations on the road network. Each CAM carries status and attribute details of the ITS-S that generated it, with its content varying based on the specific type of ITS-S, also this kind of message is sent in a continuous way to maintain real-time information. The structure of a CAM message follows a predefined format as illustrated in Figure 2.3.

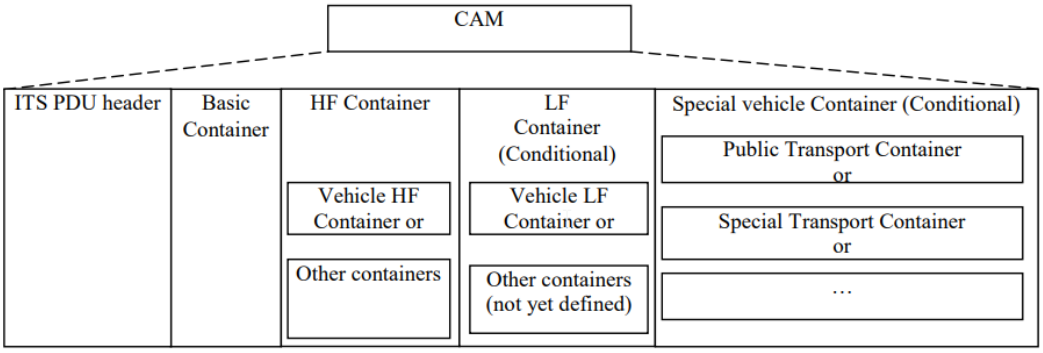


Figure 2.3: CAM structure (extracted from [14])

A description of each element can be found below:

- **ITS PDU header:** Identifies the version of the ITS message and provides basic protocol information. Its format can be found in [15].
- **Basic container:** The Basic Container provides essential information about the originating ITS station, including its type and latest geographic position at the time of message generation.
- **HF Container:** All CAMs generated by a vehicle ITS-S shall include at least a high-frequency vehicle (Vehicle HF) container. It contains high-frequency data related to vehicle dynamics such as heading or speed. The Vehicle HF container contains all fast-changing (dynamic) status information of the vehicle ITS-S, such as heading or speed.
- **LF Container (Conditional):** Provides lower-frequency data like the status of the exterior lights.
- **Special Vehicle Container (Conditional):** Used for specific vehicle types, such as public transport or special transport vehicles, and may include additional containers based on the vehicle's role.

DENM Message

The following DENM information is based on [16]. The Decentralized Environmental Notification Message (DENM) is a type of message used in ITS to notify road users and vehicles about detected hazards or unusual traffic conditions. It operates at the facilities layer, handling event-related information and ensuring it is shared among vehicles and infrastructure using ITS communication technologies.

DENM messages are used to detect and communicate various road events, such as accidents, obstacles, or hazardous weather conditions, enabling vehicles and drivers to take appropriate precautions. Unlike CAM messages, which are continuously exchanged to maintain situational awareness, DENM messages are transmitted only when necessary, with a specific purpose. The structure of this kind of message can be seen on Figure 2.4.

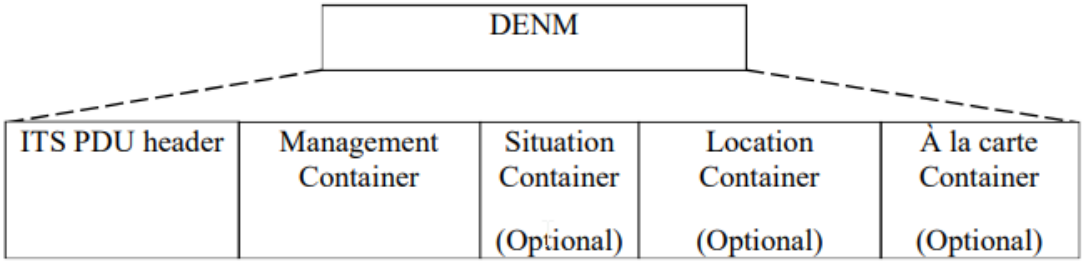


Figure 2.4: DENM structure (extracted from [16])

A description of each element can be found below:

- **ITS PDU header:** Identifies the version of the ITS message and provides basic protocol information.

- **Management Container:** Includes essential information about the event, such as detection time and event duration. This container is crucial for identifying and managing the event life cycle.

- **Situation Container (Optional):** Contains additional details about the event, describing the nature of the detected hazard or incident.

- **Location Container (Optional):** Provides detailed geographic information related to the event.

- **A la carte Container (Optional):** An extension container that allows additional, application-specific information to be included, depending on the needs of the system or service.

CPM message

The following Collective Perception Messages (CPMs) information is based on [17]. CPMs Figure 2.5 are sent by ITS-Ss to communicate information about detected objects, including vehicles, pedestrians, animals, and other potential collision hazards. This enhances the environmental perception of CPS-enabled ITS-S by providing information about non-V2X-equipped road users.

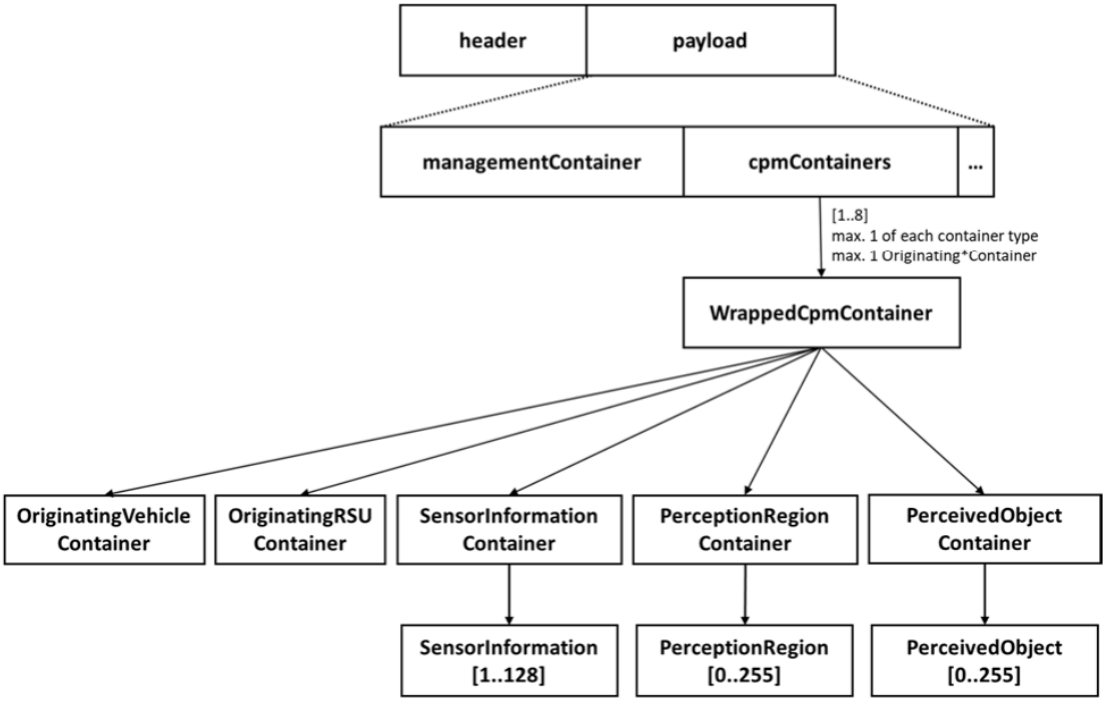


Figure 2.5: CPM Structure (extracted from [17])

A description of each element can be found below:

- **ITS PDU Header:** Identifies the version of the ITS message and provides basic protocol information.
- **Management Container:** Provides fundamental details about the originating ITS-S, such as its position and speed. It's essential to align the information with spatial and temporal context. This container is mandatory.
- **CPM Containers** can only have one Originating Container but can contain up to 7 more containers of the other types:
  - **OriginatingVehicle Container**: If the message is sent from a vehicle. It describes the originating vehicle's state focusing on the physical orientation and configuration of the vehicle..
  - **OriginatingRSU Container** If the message is transmitted by an RSU, it includes location map information.
  - **Sensor Information Container:** Details about the perception sensors used to detect objects, including sensor type and position.
  - **Perception Region Container:** Defines the field of view of the sensors, indicating the area covered by perception.
  - **Perceived Object Container:** Lists detected objects, providing details like the number of objects perceived and the classification of the objects.

VAM message

The following Vulnerable Road User(VRU) Awareness Messages (VAMs) information is based on [18]. VAM (Figure 2.6 messages are specially developed to enhance

the road safety of the class of VRUs including pedestrians, bicyclists, motorcyclists, and animals.

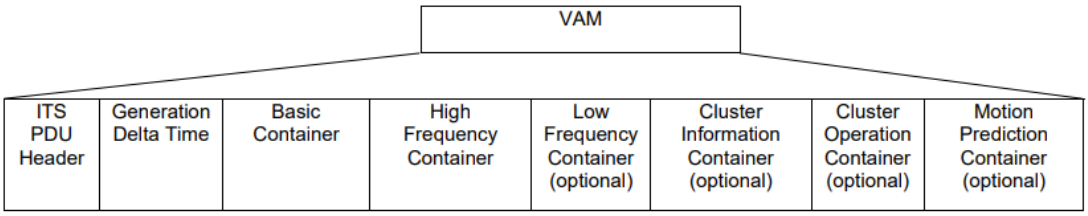


Figure 2.6: VAM Structure (extracted from [18]

A description of each element can be found below:

- **ITS PDU Header:** Identifies the version of the ITS message and provides basic protocol information.
- **Generation Delta Time:** This is a measure of the number of milliseconds elapsed since the ITS epoch,
- **Basic Container:** Provides fundamental details about the originating ITS-S, such as its position and type.
- **High Frequency Container:** Includes dynamic information such as speed and direction.
- **Low Frequency Container (optional):** Contains additional but less frequently updated attributes of the VRU.
- **Cluster Information Container (optional):** This container provides the information/parameters relevant to a VRU cluster.
- **Cluster Operation Container (optional):** Manages the joining and leaving of VRUs in a cluster.
- **Motion Prediction Container (optional):** This container provides dynamic VRU motion prediction information as well as explicit path prediction.

IVIM message

An IVIM supports mandatory and advisory road signage such as contextual speeds and road works warnings. IVIM provides information on physical road signs such as static or variable road signs, virtual signs, or road works [19].

By effectively distributing critical traffic information, the IVI service enhances road safety and optimizes traffic flow. It supports real-time decision-making by providing drivers with essential updates that influence driving behavior and route selection. This system is a key component of ITS, ensuring that vehicles remain informed and can adapt to changing road conditions and regulations. The structure of IVIMs is designed for clear and efficient communication with drivers and onboard systems. These messages typically contain elements such as geographic coordinates, validity

periods, the type of road event (e.g., construction zones, speed limit adjustments), and recommended driver actions. This structured format ensures that information is conveyed in an intuitive and easily interpretable manner, facilitating seamless integration within the ITS framework [20].

2.2.5 Services Available in Portugal

The following information was taken from [21] and it refers to the products refereed in Figure 2.2 that are currently available in Portugal. Currently, the European Commission (EU) runs the C-ITS Point of Contact (CPOC), it provides the TLM and the European Certificate Trust Lists (ECTL).

The different service elements of the Public key Infrastructure(PKI) are accessible using the following links:

Distribution Center: https://0.dc.pilot.croads.ccms.pt/

Root Certification Authority (RCA) is registered at ECTL level 0:

- RCA: https://0.dc.pilot.croads.ccms.pt/getcacerts/85987527EE188A49
- RCA-CTL: https://0.dc.pilot.croads.ccms.pt/getctl/85987527EE188A49
- RCA-CRL: https://0.dc.pilot.croads.ccms.pt/getcrl/85987527EE188A49

Certification Authorities (CA's) geographically restricted to Portugal, typically used for provisioning Road Side Units (RSU's):

- EA-PT: https://0.ea.pilot.croads.ccms.pt/getcacerts/BA12069B91B8E810
- AA-PT: https://0.aa.pilot.croads.ccms.pt/getcacerts/9047B112864EDDFD

CA's with no geographic restrictions, typically used for provisioning On Board Units (OBU's).

- EA-WW: https://0.ea.pilot.croads.ccms.pt/getcacerts/CAC1A29FA18FAF14
- AA-WW: https://0.aa.pilot.croads.ccms.pt/getcacerts/369F636239546522

Generic CA's, used for testing purposes.

- EA-TS: https://0.ea.pilot.croads.ccms.pt/getcacerts/4C87B1B09B8CD364
- AA-TS: https://0.aa.pilot.croads.ccms.pt/getcacerts/0646294BB7BE3D8F

Main public endpoints for security provisioning of C-ITS stations:

- https://0.ea.pilot.croads.ccms.pt/ec
- https://0.aa.pilot.croads.ccms.pt/at

2.3 Related Work

The European C-Roads platform and C-Streets project have been significant initiatives aimed at deploying interoperable C-ITS services across Europe. These projects focus on standardization, large-scale deployment, and ensuring cross-border interoperability. Their objectives were discussed in Section 1.2.

In the hybrid communication solutions domain, several studies have explored the integration of message-oriented middleware in C-ITS. For instance, the objective of [22] study is to assess the feasibility of integrating MQTT and Apache Kafka through a bridging architecture to support C-ITS. Specifically, the study aims to determine whether such integration can ensure interoperability between MQTT-based mobile/vehicular applications and Kafka-based data processing systems, to evaluate the system's performance in terms of latency, message delivery, and scalability.

Additionally, [23] presents a design and evaluates a vehicular communications framework that integrates roadside infrastructure with cloud-based services to support the delivery of Cooperative Intelligent Transport Systems (C-ITS) applications. The study aims to ensure low-latency and reliable communication at the roadside, while leveraging the cloud's scalability and processing capabilities to enhance traffic management, road safety, and cooperative mobility services.

The Authors in [24] propose and experimentally validate a hybrid vehicular communication mechanism that jointly employs short-range technologies, such as ITS-G5, and long-range cellular networks, such as LTE, for wireless vehicular networks. The objective is to satisfy the diverse temporal and spatial requirements of connected and autonomous vehicle applications. The study aims to characterize the performance trade-offs between ITS-G5 and LTE when used independently, considering metrics such as latency, throughput, and reliability, and to demonstrate that the hybrid integration of these technologies can better meet the requirements of C-ITS than either technology alone.

The Thesis in [25] addresses security management in vehicular networks by analyzing existing standards and proposing an active revocation algorithm for pseudonym certificates, namely Authorization Tickets (ATs). Unlike the passive revocation approach, the proposed scheme relies on an Authorization Ticket Certificate Revocation List (ATCRL) actively distributed across the network, enabling vehicles to immediately discard messages from compromised stations. Both centralized and decentralized variants of the algorithm were evaluated under different scenarios, demonstrating its potential to reduce the vulnerability window inherent to passive revocation mechanisms. While the present work focuses primarily on end-to-end latency analysis, studies such as this highlight the complementary role of robust security mechanisms in ensuring trustworthy vehicular communications alongside performance considerations.

Finally, projects like PASMO,[26], provide an open living lab for the development and validation of Cooperative Intelligent Transport Systems (C-ITS) and smart regions, integrating not only vehicles and roadside units but also a wide range of devices such as cameras, sensors, and digital services. This heterogeneous environment enables

experimentation in realistic conditions and underlines the importance of assessing the performance of C-ITS solutions as complete systems. In this thesis, we adopt a similar perspective by analyzing the end-to-end latency of a hybrid C-ITS system deployed by IP.

2.4 Technical Challenges

This section presents the main technical challenges encountered throughout the project, starting with the architecture itself, followed by the implementation of MQTT, and concluding with the development of the mobile application. One of the most significant challenges arises from the fact that this is a pilot implementation, and therefore, the system is not yet robust against errors.

2.4.1 Tiling Structure

In the hybrid model, legacy devices receive C-ITS information by subscribing to MQTT topics associated with their physical location, following a tiling structure illustrated in Figure 2.7. Each tile corresponds to a dedicated MQTT topic, ensuring geographically structured and location-specific dissemination. Devices within a given tile subscribe only to its corresponding topic, thus avoiding unnecessary message reception.

As a device moves, it must continuously monitor its position and update its subscriptions when transitioning between tiles. This requires unsubscribing from the previous topic and subscribing to the next one, enabling seamless data reception along its trajectory. Near tile boundaries, devices may temporarily subscribe to two topics simultaneously to ensure continuity. However, this introduces the risk of duplicate messages, which must be properly managed to prevent redundancy and inconsistencies.

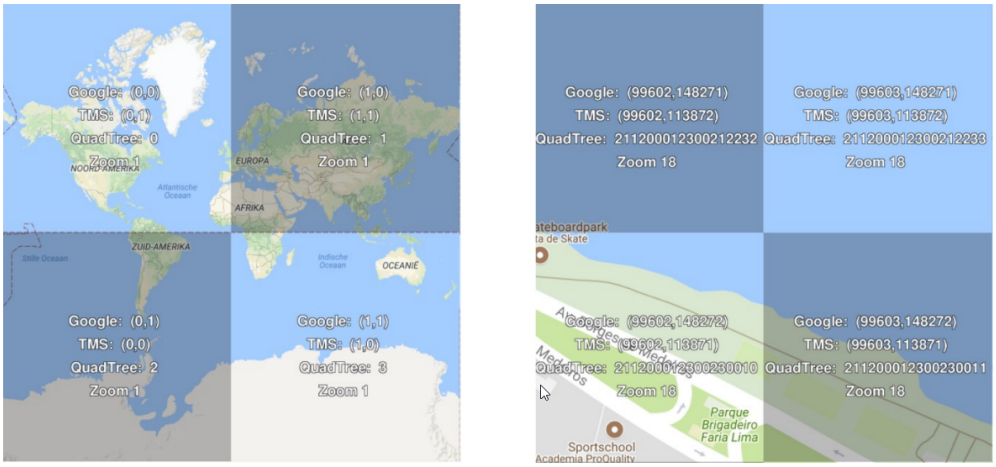
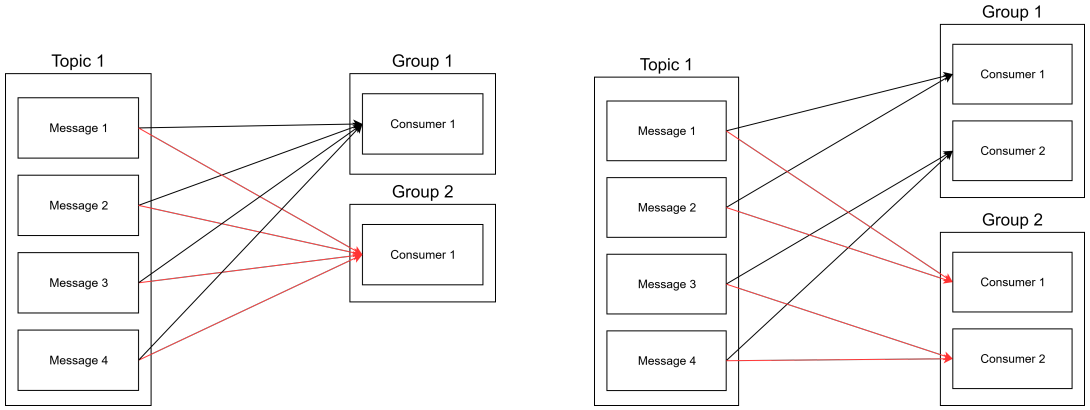


Figure 2.7: Tiling concept with less zoom on the left and more on the right (adapted from [27])

2.4.2 MQTT Implementation

A critical challenge in this project is ensuring reliable message delivery within the MQTT-based communication system. A key requirement is that every subscriber to a given topic must receive all messages, avoiding any repartitioning of messages across different consumers.

MQTT can operate in two distinct ways. In Figure 2.8a, the desirable behavior is illustrated: if there is a queue of messages, each message should be delivered to all subscribers. Conversely, Figure 2.8b shows an undesired behavior, where messages are repartitioned across different consumer groups. In this case, subscribers in Group 1 and Group 2 would each receive only a subset of the messages, meaning that no group has full visibility of the complete message set.



(a) MQTT without Partition (b) MQTT with Partition

Figure 2.8: MQTT Message Distribution

2.4.3 Mobile Application

For more efficient topic management, an important feature and a relevant challenge would be the ability to subscribe to all relevant topics along a planned route. For example, if a user selects a route from Lisbon to Porto, the application should automatically subscribe to all MQTT topics covering that path. This guarantees continuous reception of real-time road information, providing timely alerts about traffic disruptions or hazards that may affect the journey.

3
C-ITS Architecture Overview

This chapter focuses on a real-world architecture currently implemented by *Infraestruturas de Portugal* (IP). It begins with a detailed overview of the system's structure, describing each component and its specific function. Next, performance tests will be carried out to measure delays between components and identify potential areas for improvement. Finally, a resilience test will be conducted to assess how the system behaves under edge-case scenarios, such as component failures.

Since one of the primary challenges of this project is evaluating a suitable architecture, given the absence of an officially established model, the structure illustrated in Figure 3.1 will serve as a reference.

All components are deployed as independent Docker containers, which ensures modularity, environment consistency, and ease of deployment across the architecture. This containerized approach allows each component to operate in isolation, avoiding dependency conflicts and enabling updates or restarts without affecting the remaining services.

The red parts of the architecture represent elements that are planned but not yet fully functional; these will be explained in 3.4.

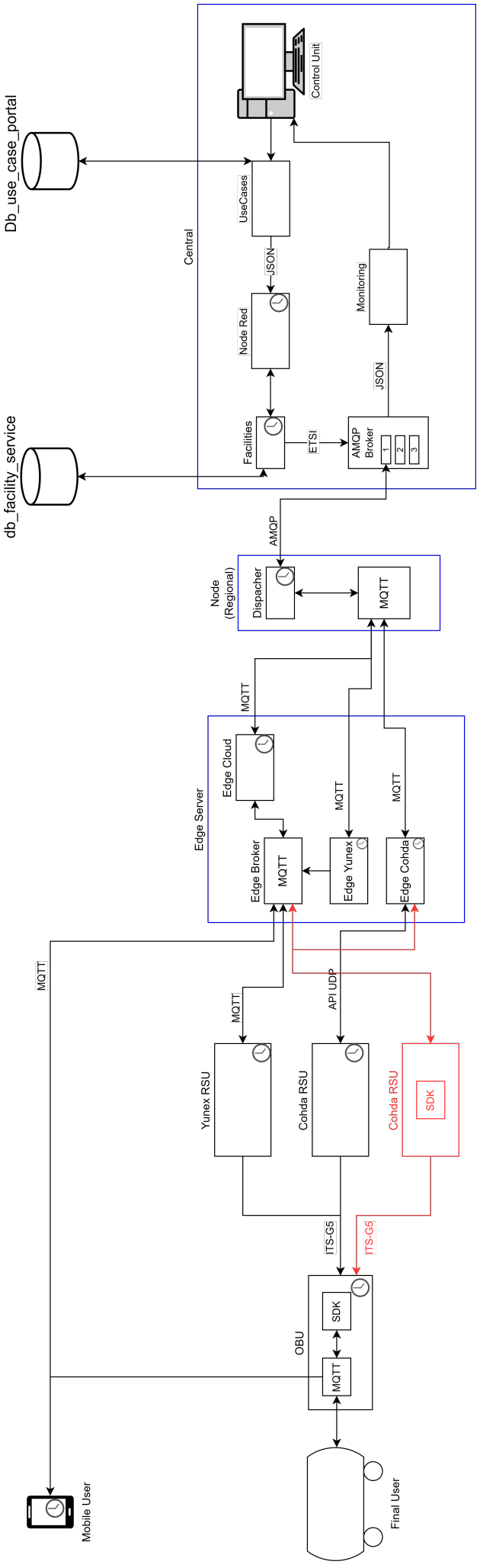


Figure 3.1: IP Architecture

3.1 Central Unit

Starting from the right side of the schema, we have the Central Unit (Figure 3.2) that acts essentially as the brain of the system.

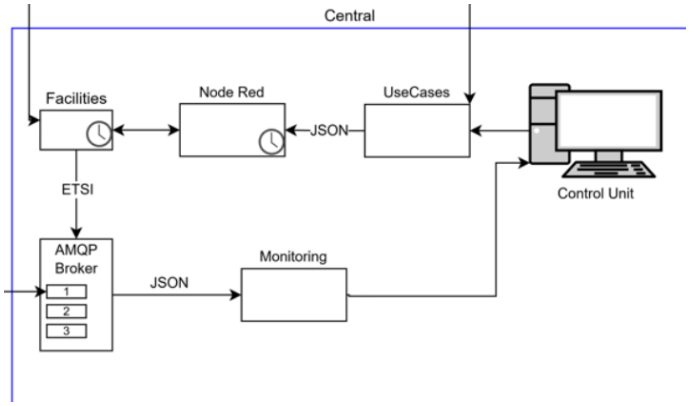


Figure 3.2: Central Unit

Control Unit

Within the Central Unit, the Control Unit is implemented as a web service that provides the connection between the user and the system through a graphical user interface (GUI).

The Management tab of the GUI is shown in Figure 3.3. This tab displays the RSU equipment available for transmission, as well as the units currently unavailable to the system and their associated warnings, such as high temperatures. On the left side, the names of the edges are visible. These correspond to the components represented in Figure 3.1 within the ”Edge Server” component as Edge Cloud, Edge Yunex, and Edge Cohda. Their purpose is detailed in Section 3.3.

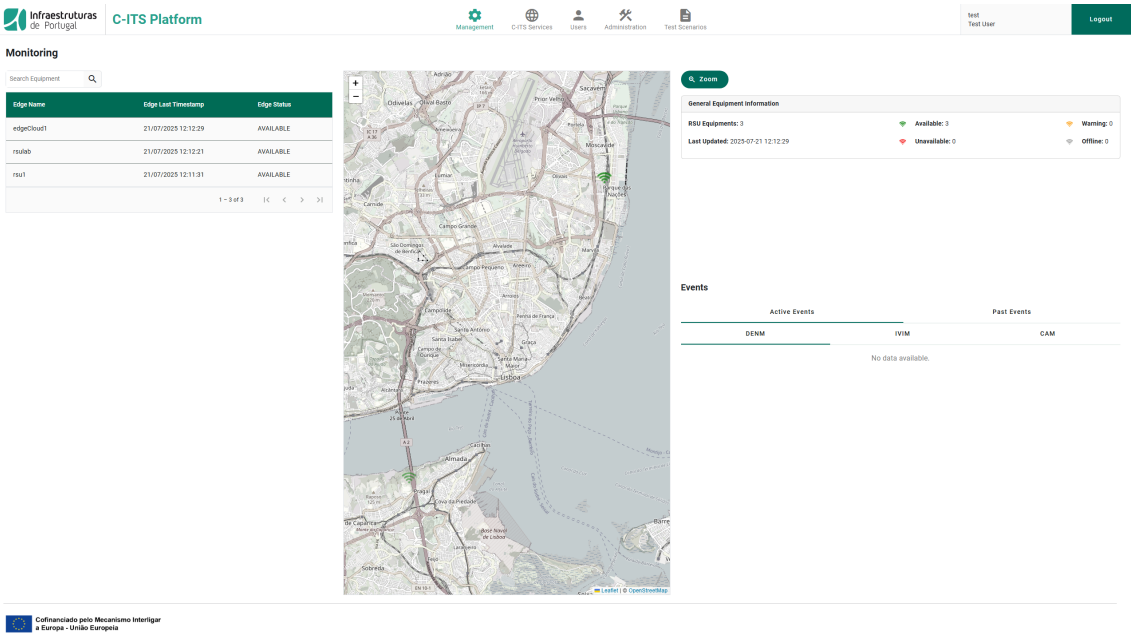


Figure 3.3: API Connection to Mobile APP

It is also possible to click on these edges to view the information shown in Figure 3.4.

One of the most essential pieces of information shown in Figure 3.4 is the list of all current active events being transmitted by the RSU, along with the RSU status, indicating whether the physical equipment is operational or not.

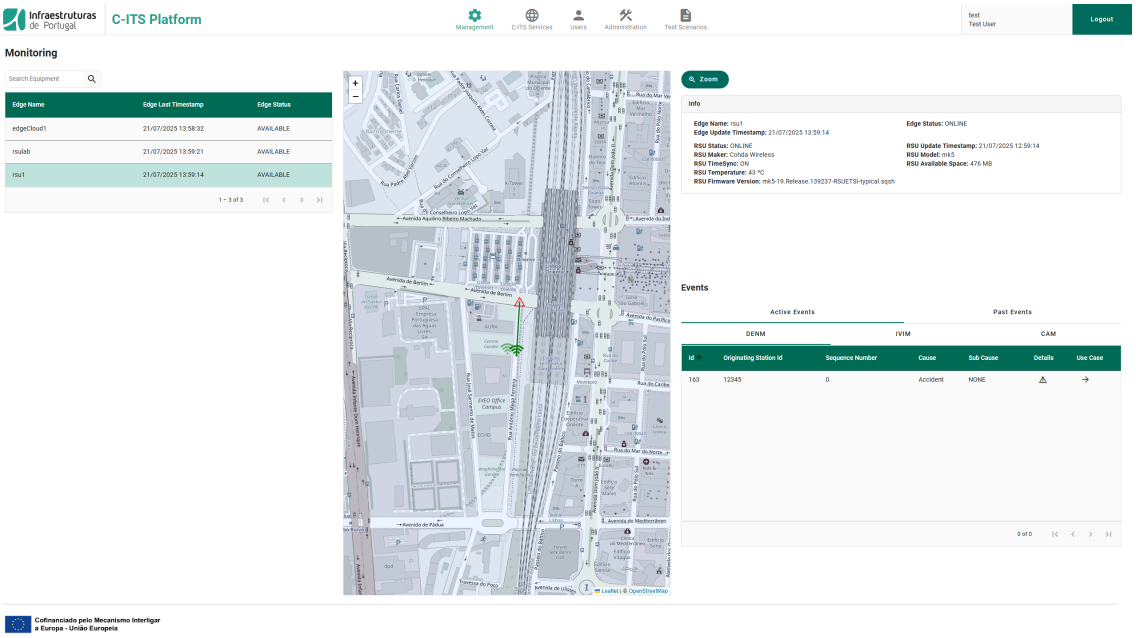


Figure 3.4: Current Active Events

The C-ITS Services tab of the GUI is shown in Figure 3.5. This is where events of types DEN and IVI can be created, allowing them to be sent to the RSU, which will then forward them to the OBU. It also provides a view of the currently active events and their location.

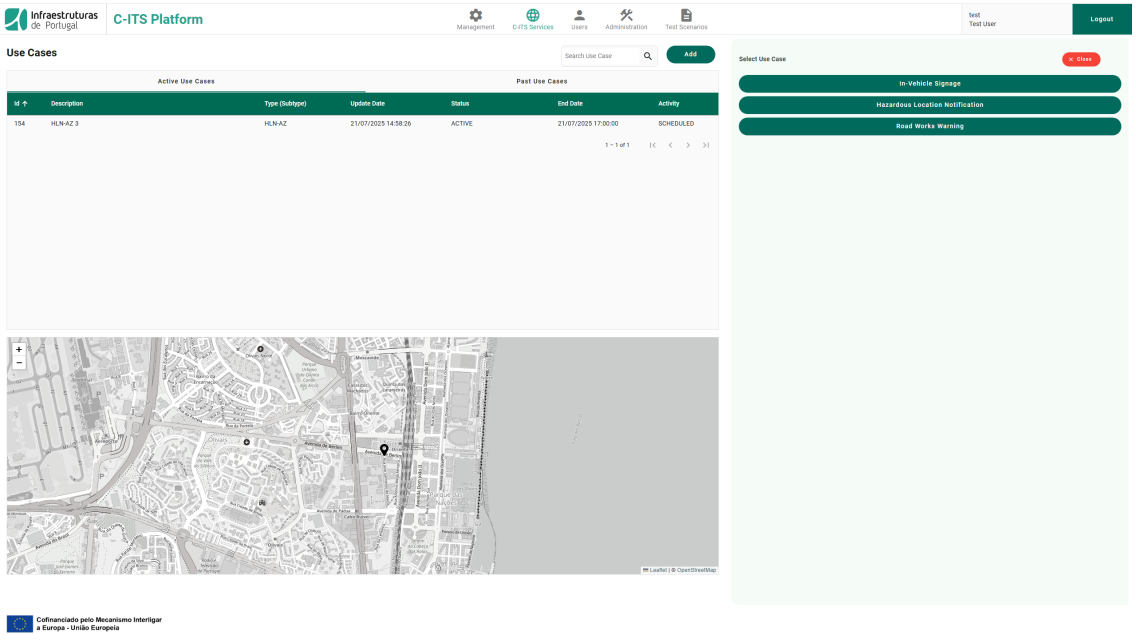


Figure 3.5: C-ITS Services Tab

Inside each category, several use case types are presented. For instance, in the case of Hazardous Location Notification (HLN), the available options are shown in Figure 3.6.

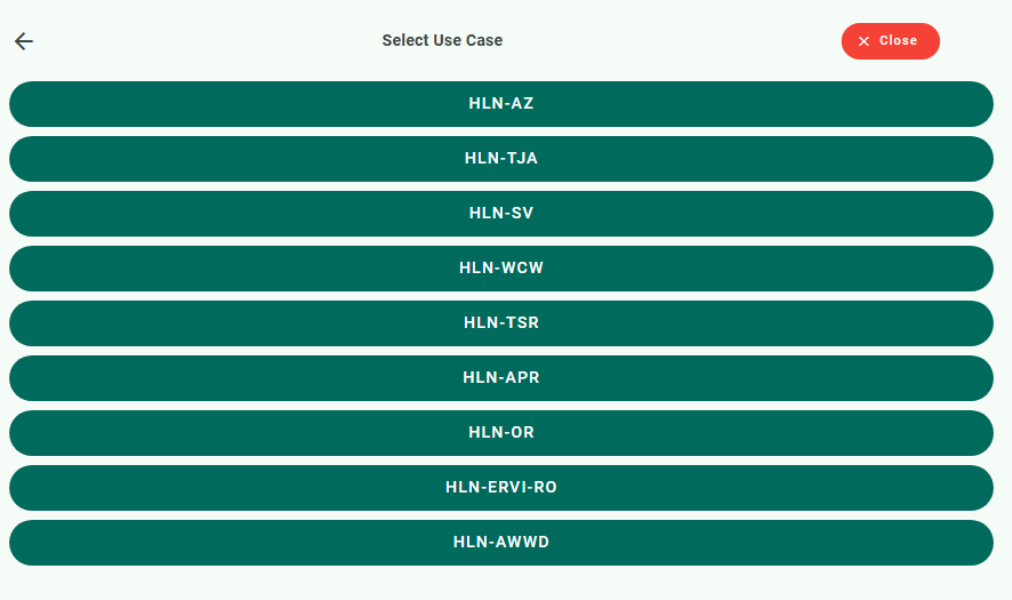


Figure 3.6: C-ITS Use Cases

After selecting a case and completing the required information, a JSON file is generated and sent to the Use Cases service, which processes the event according to its type.

The remaining tabs shown in Figure 3.3 are not described, as they were still under development and not functional at the time of this project.

Use Cases

The Use Cases Microservice collects all the parameters defined by the user when creating a message in the Control Unit and structures them into a JSON file, ensuring compatibility with Node-RED for interpretation. In addition, the service stores all transmitted messages in its database, enabling later retrieval and consultation.

Node-Red

This service processes JSON messages generated by the Use Cases microservice, extracting all relevant data based on message type to construct the ETSI-compliant format. The extracted fields are then forwarded to the Facilities Service. Due to confidentiality constraints, the internal logic of the Node-RED flow cannot be disclosed.

Facilities

The Facilities Service receives all the information extracted by the Node-RED flow and is responsible for completing the ETSI message using those data fields. It includes a PostgreSQL database to store all generated messages. At the final stage, the

service communicates with the Regional Node through an AMQP (Advanced Message Queuing Protocol) broker. AMQP was chosen over MQTT to ensure reliable message delivery, as guaranteed transmission to the Node is a critical requirement.

Monitoring

The feedback loop from the physical RSUs, including temperature information and warnings, to the Central Unit, does not pass through the Facilities component shown in the diagram. Instead, it is handled by a dedicated microservice called Monitoring, which is responsible for receiving and processing this type of message.

3.2 Regional Node

In this architecture, the Node (Figure 3.7) serves as a means of distributing workload across multiple components. The goal is to deploy several Nodes throughout the country, each subscribed only to the AMQP topic relevant to its geographic area and optionally to a general topic for nationwide information. Inside each Node, there is a service called Dispatcher, which, as its name suggests, is responsible for receiving messages from AMQP and forwarding them to the corresponding topic within the locally deployed MQTT broker.

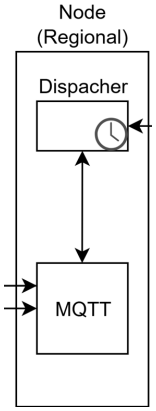


Figure 3.7: Regional Node

3.3 Edge Server

The Edge Server (Figure 3.8) is responsible for bridging the physical components with the digital services described earlier in this chapter. All components within the Edge Server run as Docker containers, except for the Edge Broker, which operates using MQTT technology.

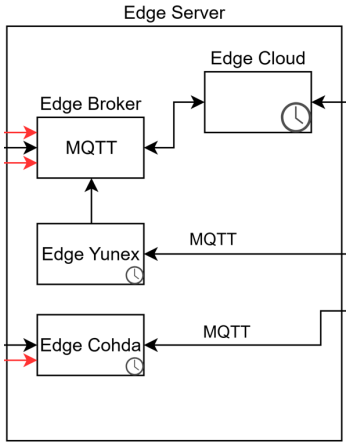


Figure 3.8: Edge Server

Edge Cloud

This service was designed to act as a nationwide Virtual RSU, aggregating all messages transmitted within the C-ITS infrastructure. All Regional Nodes forward their messages to this central component, making it the single point where all C-ITS data converge. The purpose of this architecture is to provide a unified environment for mobile application developers, allowing them to consume messages in real-time. The responsibility for distributing these messages to relevant areas and users lies entirely with the applications themselves. This should be the only component interfacing directly with external mobile applications. A typical example of such an application could be Waze.

Edge Broker

The Edge Broker is the MQTT service responsible for receiving messages from physical RSUs, including temperature readings and device-generated warnings. It also handles the transmission of messages originating from the Yunex Edge, and potentially from the Edge Cohda in the future. Message direction is managed through distinct MQTT topics. Currently, messages from the Cohda follow a separate transmission path, which will be explained in the following sub-chapter.

Edge Cohda

Edge Cohda is a Container based service that ”manages bidirectional transmission by the RSU in accordance with the defined ETSI standards and the C-ROADS profile” as explained in [28]. It is essential to note that all RSUs transmit status information every second, including temperature, location, and other metrics, which is received by the Edge Server. The Edge Server acts as a data bus, meaning it filters out repeated data and only forwards messages that represent a change, which are therefore considered relevant. When such a change is detected, the information is sent back to the Control Unit, which displays it to the user.

Edge Yunex

Similar to the Edge Cohda, this component captures the messages published on the MQTT broker present in the Regional Node and forwards them to the Cohda physical device. Due to specific implementation constraints of the device, this communication is carried out using UDP. Like the Edge Cohda, this component also acts as a filter for warning messages sent from the RSU, functioning in a similar way to the Edge Yunex component.

3.4 C-ITS Physical Equipments

This sub-chapter analyses the physical components (Figure 3.9) and explains their roles within the overall architecture, detailing their specific purposes. These components are fundamental to the interaction between the physical infrastructure and the digital services. Understanding their behavior is essential to grasp the overall functioning of the system.

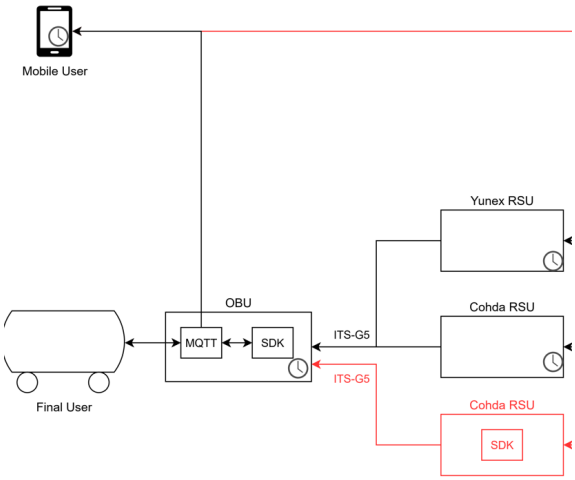


Figure 3.9: Physical Equipments

RSU & OBU

On the RSU layer, two different manufacturers are involved: Cohda (as shown in Figure 3.10a) and Yunex (as shown in Figure 3.10b), each adopting distinct operational approaches.

(a) Cohda OBU (extracted from [29])

(b) Yunex RSU (extracted from [30])

Figure 3.10: OBU and RSU used in the tests

In the case of Cohda, for lab purposes, an OBU is being used as an RSU. This is possible because Cohda devices can operate as either RSU or OBU, depending on the configuration and inputs provided to the software running on the device. The Yunex RSU supports direct connection to an MQTT broker, which is the method currently in use due to its simplicity and efficiency.

On the other hand, the Cohda RSU does not natively support MQTT. Instead, it communicates using UDP, which introduces a divergence in the system architecture. Ideally, all RSUs would connect directly to the Edge Broker via MQTT to ensure uniformity. As previously mentioned, both Cohda devices can operate as RSUs or OBUs. Therefore, a near-future goal of the company implementing this architecture is to adapt the SDK currently used on the OBU and deploy it on the RSU, enabling direct MQTT communication with the Edge Broker, this is represented in red in Figure 3.1.

On the OBU side, communication from the RSU is carried out using ITS-G5. All OBUs installed are Cohda devices. As referred before, using the SDK, the company developed a solution to enable MQTT communication on the OBU. The MQTT broker runs locally on the device and is used to forward all received messages to the mobile application.

Mobile Application

The mobile application is also a component of the system, although it was developed only for testing purposes and will never be available for the final user, serving as the interface through which the results of the transmitted messages can be visualised on a real map. It is currently connected to an MQTT broker running locally on the OBU. However, the long-term objective is to connect the application to the Edge Cloud component presented in 3.3.

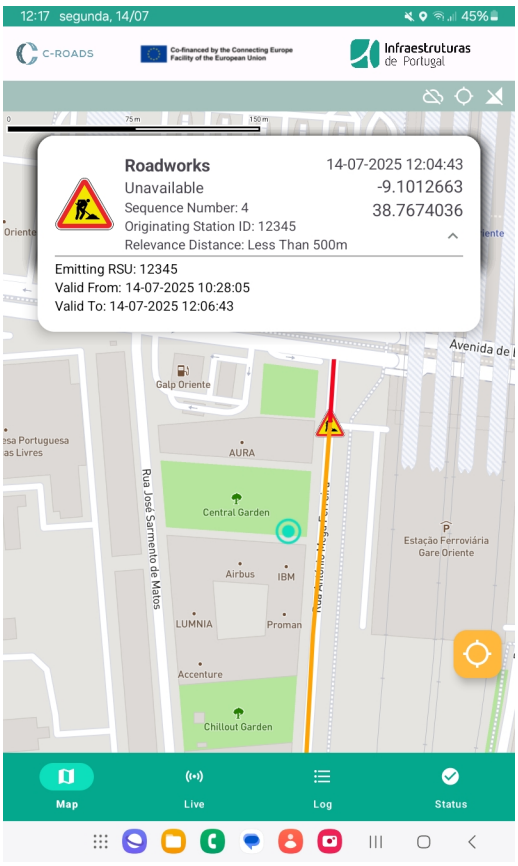


Figure 3.11: Testing Mobile App

The application is shown in Figure 3.11, where a RoadWork message has been received, including its central coordinate. On the map, the danger zone is represented in red, while the warning zone is shown in orange. This means that any vehicles within the warning zone will be alerted about the roadwork occurring in the danger zone.

Additionally, the application displays the RSU that transmitted the specific message, along with its validity period, indicating both the start and end times during which the message remains valid.

On the Live tab, all real-time events currently being transmitted by the OBU are displayed.

On the Log tab, it is possible to view the application logs, such as the topics to which the application is subscribed.

Finally, the Status tab allows the user to define the MQTT broker IP address from which the application will consume messages.

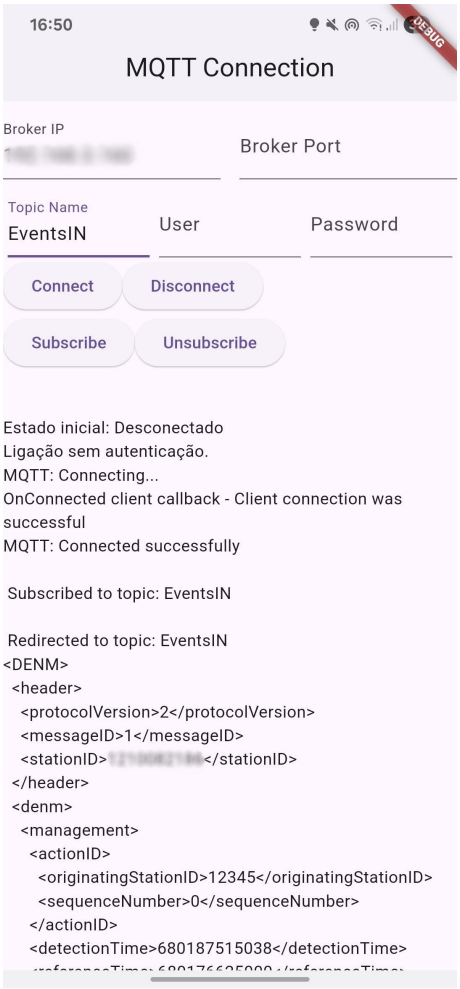


Figure 3.12: Developed APP Connected to Industrial Architecture

In Figure 3.12, an application developed for this project is shown connected to the industrial implementation. This demonstrates that, with the proposed architecture, third-party users can connect to the MQTT broker and receive all information distributed on the road. It also enables any individual user to develop a personal application to access traffic information, as well as allowing companies to integrate this data into their infrastructures.

4

Industrial Architecture Evaluation

This chapter presents the evaluation of the architecture introduced in 3. It begins with an overview of the setup used for the tests, followed by the delay measurements between components, where the methodology and results are explained, and finally, the resilience tests are described together with their corresponding results.

4.1 Used Setup

In this section, the physical setup used to test the system is described. Two diagrams are presented to support the explanation and illustrate the physical connections, followed by pictures of the actual setup, providing a direct view of the implemented connections. In Figure 4.1 and Figure 4.2, the connection schemes used to access the RSU and OBU are presented.

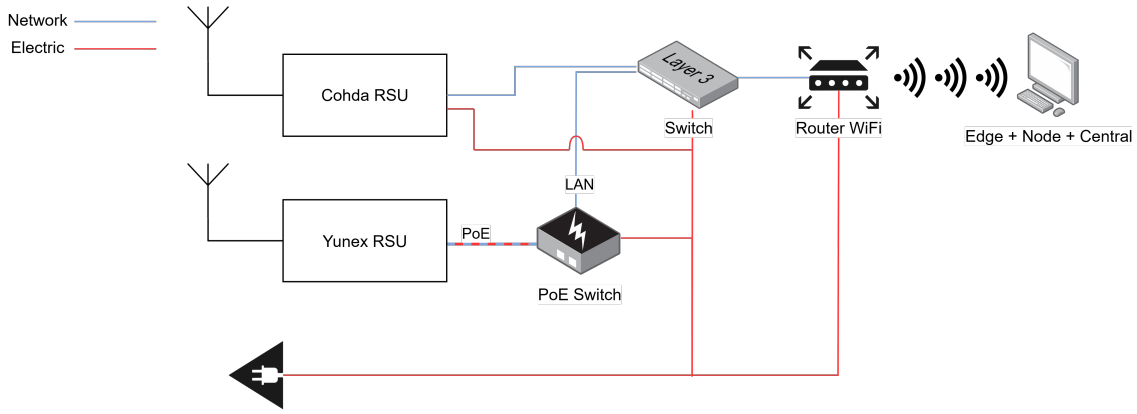


Figure 4.1: RSU Physical Setup Diagram

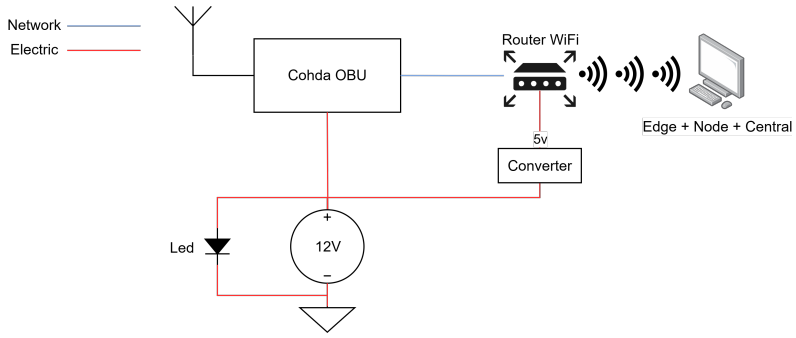


Figure 4.2: OBU Physical Setup Diagram

On the RSU side, the two devices are connected in different ways. The Cohda RSU is connected directly to a switch, which is then linked to the Wi-Fi router through an RJ45 cable, while its power supply is provided directly from the source. In contrast, the Yunex RSU is powered through PoE, which simultaneously provides energy and network connectivity to the router via a switch. The PC then accesses the internal Wi-Fi network created by the router, obtaining access to the devices without requiring a direct cable connection, as it would be in the original method.

On the OBU side, a diagram is presented showing how it is possible to connect to the OBU without the need for a physical RJ45 connection, similar to the RSU setup. In this case, the connection is simpler, with the OBU connected directly to the router, which then provides access to the OBU through its internal Wi-Fi network.

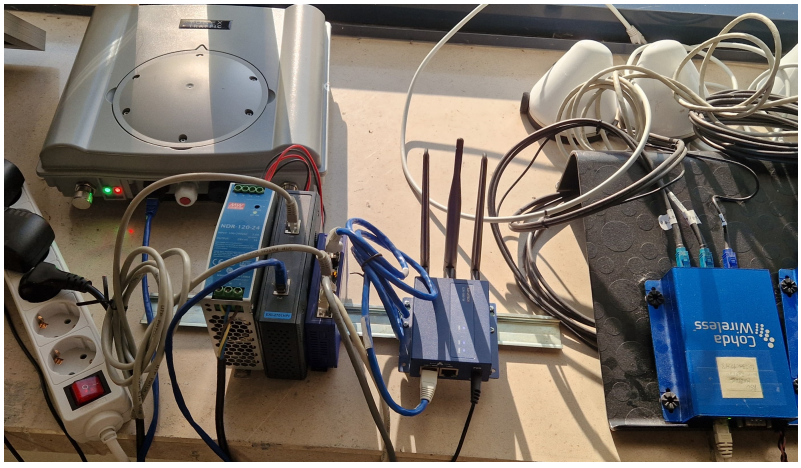


Figure 4.3: RSU Physical Setup

For illustrative purposes, Figure 4.3 shows the actual RSU setup. From left to right, the components are arranged as follows: the Yunex RSU, the power supply, the PoE injector, the switch, the router, and finally the Cohda RSU. This system is portable provided that an electrical connection is available.

Figure 4.4: OBU Physical Setup

In Figure 4.4, it can be seen that the OBU system components are contained within a single box. This configuration also makes the unit portable as long as an electrical connection is available. The router presented in Figure 4.2 is located beneath the OBU, and its antennas are visible. This one is similar to the one shown in Figure 4.3.

4.2 Delay between Components

In this section, the delays between all components of the architecture presented in Figure 3.1 will be measured. The clocks shown on various components in the figure indicate the points where timestamps are collected to measure the intermediary delays.

In this chapter, the results of the delay measurements between components are analyzed. The Use Cases component was excluded from the evaluation due to the negligible impact of its delay on the overall system. Additionally, omitting this component simplifies the process of inserting messages into the system. Including it in the delay measurements would require all messages to be created manually, following a repetitive and exhaustive procedure. By bypassing it, messages could be inserted directly into Node-RED, enabling the generation of multiple messages more efficiently using the process explained in 4.2.1.

It is essential to note that the primary objective of the end-to-end communication, from the moment the message leaves the Control Unit until it reaches the OBU, is to maintain a total delay of under 100 ms.

4.2.1 Procedure for Measuring Latency

In this subsection, the measurements carried out across the different components are explained, detailing the strategy adopted and the specific considerations taken into account for each component. The chosen approach for testing the system was based on two main factors: the ease of inserting timestamp messages into Docker containers, and the fact that all containers were running on the same machine, ensuring synchronized clocks and thus allowing direct comparisons. It is important to note that timestamp messages were not originally implemented at the beginning of this project. Their introduction was specifically encouraged and integrated during the course of this work.

Central Unit

Starting from the Control Unit, a valid JSON message is inserted into the inject block named "Injetar 'teste'". This block injects the message into the function "Repetir 100x", which increments a counter up to 100 on each execution. A configurable delay can also be introduced between iterations. The corresponding Node-RED block group is shown in Figure 4.5.

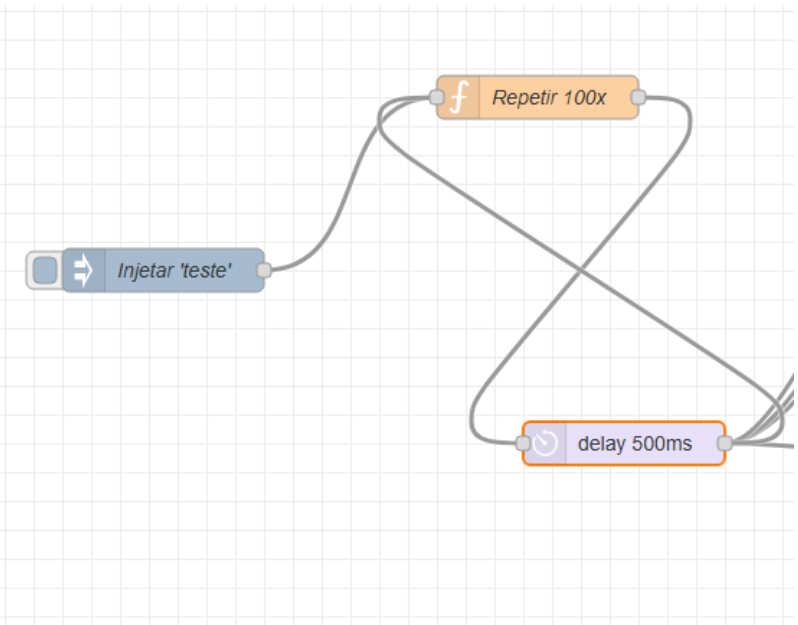


Figure 4.5: NodeRed Loop

To measure the time between components within Node-RED, a simple function was created to log timestamps. This function can be inserted at any step of the Node-RED flow to mark the time of execution, its blocks can be seen on Figure 4.6.

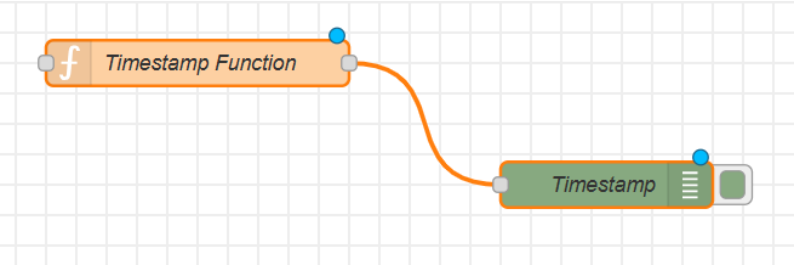


Figure 4.6: NodeRed Timestamp

The Timestamp function uses the following code to obtain the exact date and time, formatting it into a user-friendly string.

o que é isto?

```
let now = new Date();

let hours = now.getHours().toString().padStart(2, '0');
let minutes = now.getMinutes().toString().padStart(2, '0');
let seconds = now.getSeconds().toString().padStart(2, '0');
let milliseconds = now.getMilliseconds().toString().padStart(3, '0');

msg.payload = `${hours}:${minutes}:${seconds}.${milliseconds}`;
return msg;
```

Inside the Node-RED flow, the final step is a web call to the Facilities API. For this reason, the timestamp block is also executed at the start of the Facilities process call and again upon its return, marking both the end of the Node-RED process/beginning of the Facilities process, and the end of the Facilities process, which corresponds to the conclusion of all measurements taken within the Central Unit.

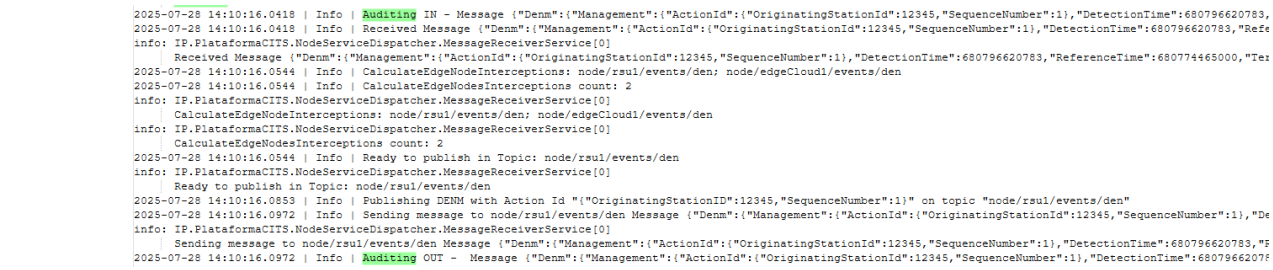
To process this information, a script was developed to extract all data from a document containing the output from the Node-RED container console. The script filters all messages printed in the format generated by the debug function and organizes them into a ".txt" file, following the structure presented in Figure 4.7. The labels "Before ETSI" and "After ETSI" indicate the beginning and end of the Facilities process call, respectively.

Main Entry	Before ETSI	After ETSI
14:08:34.198	14:08:34.560	14:08:35.762
14:10:14.965	14:10:15.019	14:10:15.854
14:11:58.898	14:11:58.915	14:11:59.388
14:11:59.400	14:11:59.409	14:11:59.839
14:11:59.901	14:11:59.911	14:12:00.365
14:12:00.404	14:12:00.419	14:12:00.883
14:12:00.905	14:12:00.918	14:12:01.340
14:12:01.406	14:12:01.417	14:12:01.826
14:12:01.907	14:12:01.917	14:12:02.299
14:12:02.408	14:12:02.418	14:12:02.874
14:12:02.909	14:12:02.920	14:12:03.410

Figure 4.7: NodeRed Filtered Times

Node

In the Node component, it was possible to insert a timestamp both when a message arrives and when it leaves the Dispatcher container. This makes it possible to measure the time taken from the moment the message leaves the Facilities until it reaches the Dispatcher, as well as the time the Dispatcher takes to process it. The logs of incoming and outgoing messages are stored in the container's log console, and an example can be seen in Figure 4.8.



demasiado pequeno para ler

Figure 4.8: Dispatcher Auditing Messages

As it would be exhausting to retrieve these messages one by one, a script was developed that, similarly to the Central Unit script, filters the relevant messages and organizes them into a separate file. The messages, already ordered by their sequence number, can be seen in Figure 4.9.

IN	OUT	SequenceNumber
14:10:16.0418	14:10:16.0972	1
14:11:59.4362	14:11:59.4362	2
14:11:59.8865	14:11:59.8865	3
14:12:00.4285	14:12:00.4285	4
14:12:00.9368	14:12:00.9431	5
14:12:01.3869	14:12:01.3869	6
14:12:01.8790	14:12:01.8790	7
14:12:02.3727	14:12:02.3737	8
14:12:02.9146	14:12:02.9146	9
14:12:03.5640	14:12:03.5640	10
14:12:04.0323	14:12:04.0338	11
14:12:04.4058	14:12:04.4058	12
14:12:04.8728	14:12:04.8738	13

Figure 4.9: Dispatcher Filtered Times

Edge Server

Inside the Edge Server, it is possible to retrieve timing information from three different points: within the Edge Cloud, indicating the moment when the architecture makes the messages available to the mobile user, as well as the arrival and departure times for both the Edge Yunex and Edge Cohda. The arrival times indicate how long it takes for the messages to pass through the MQTT broker and reach the Edge components, while the departure times show how long these components take to process the messages and make them available to the physical equipment.

The process of obtaining the timing information is similar to that used in the Dispatcher, as these are also containers and support logging. Similarly to what was

done in the Dispatcher container, a script was created to filter the relevant messages from the rest of the logging information. The filtered times can be seen in Figure 4.10.

Data	Hora_IN	Hora_OUT	SequenceNumber
2025-07-28	14:10:16.1106	14:10:16.2368	1
2025-07-28	14:11:59.4401	14:11:59.4432	2
2025-07-28	14:11:59.8923	14:11:59.8948	3
2025-07-28	14:12:00.4426	14:12:00.4431	4
2025-07-28	14:12:00.9454	14:12:00.9454	5
2025-07-28	14:12:01.3928	14:12:01.3930	6
2025-07-28	14:12:01.8822	14:12:01.8838	7
2025-07-28	14:12:02.3759	14:12:02.3759	8
2025-07-28	14:12:02.9185	14:12:02.9185	9
2025-07-28	14:12:03.5680	14:12:03.5680	10
2025-07-28	14:12:04.0369	14:12:04.0369	11

Figure 4.10: CohdaEdge Filtered Times

Physical Components

The last point where time measurements are taken is at the physical equipment level, specifically inside the RSU and OBU devices. Both Cohda and Yunex units can store PCAP files of the transmitted (TX) and received (RX) messages. By analysing these PCAP files, it is possible to extract timing information. On the RSU side, the TX file is analysed, while on the OBU side, the RX file is used.

An example of the TX PCAP file inside the RSU is shown in Figure 4.11, where the Arrival Time indicates the exact moment at which the message reached the RSU.

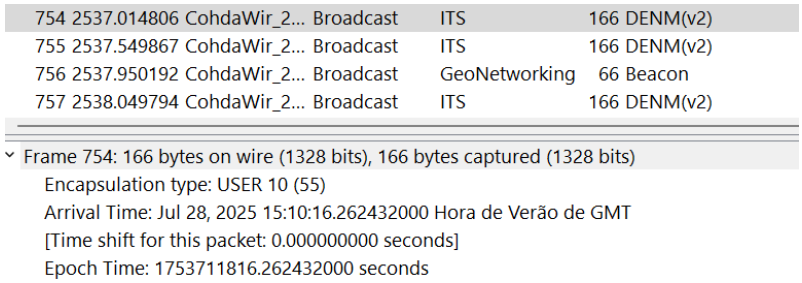


Figure 4.11: RSU Tx PCAP File

In the same PCAP file, the sequence number of the message can be observed, as shown in Figure 4.12. This identifier makes it possible to distinguish between different messages and to compare them with those carrying the same sequence number in the other components.

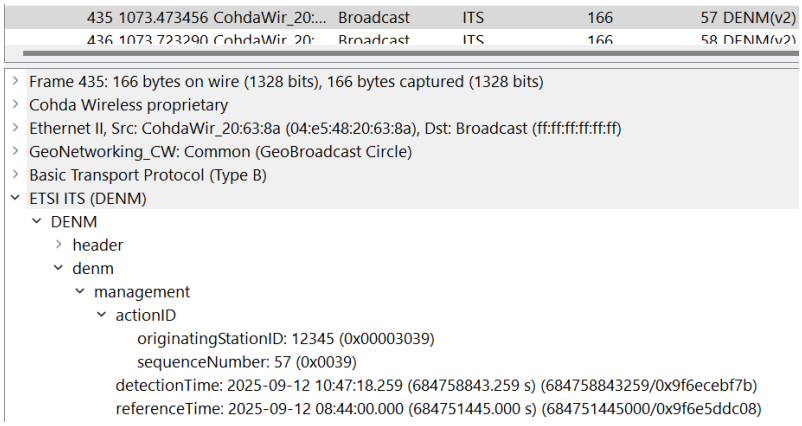


Figure 4.12: PCAP Sequence Number

4.2.2 Load Tests

In this chapter, load tests are performed with three main objectives: first, to evaluate the system's capacity to withstand a high volume of messages; second, to reliably identify where most of the delay is introduced; and finally, to determine the actual average end-to-end latency of a message within the system.

Single Message

When sending a single message through the system and measuring the processing times across the different components, it is difficult to draw definitive conclusions, as several factors—such as network load at different times of the day—may influence the results. Nevertheless, this test was carried out, resulting in an end-to-end time of 0.830 seconds. The analysis of the intermediate times is presented in Figure 4.13.

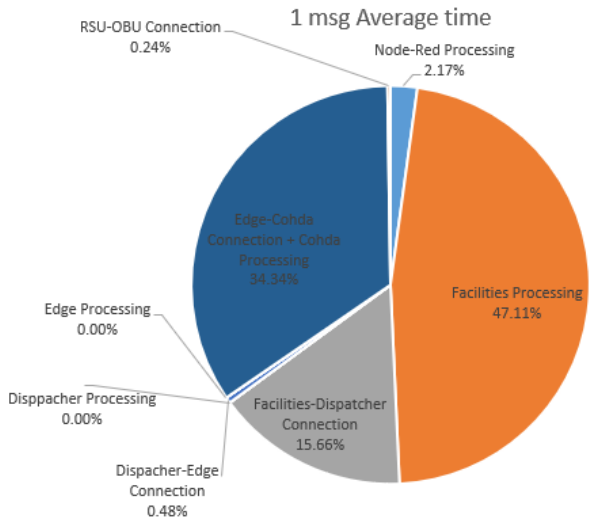


Figure 4.13: Components Delays 1 Message

Two types of delays can be distinguished: connection delays and processing delays. The processing delay corresponds to the time required for a container to

process the provided data, while the connection delay corresponds to the time it takes for a message to travel from one component to another.

In Figure 4.13, three main components can be identified as the primary sources of delay, represented by the orange, grey, and blue slices. Among them, the orange slice—corresponding to Facilities Processing—accounts for nearly 50% of the total delay. The grey slice represents the time required for a message to travel from the Facilities container to the Dispatcher container, considering the AMQP component as the intermediary connection between these two elements, and accounts for 15.6% of the delay. Finally, the blue slice corresponds to the time elapsed from the moment a message leaves the Cohda container until it exits the Cohda RSU physical equipment, accounting for 34.3% of the delay.

300 Messages With 500ms Delay

Although a single-message measurement was performed, as previously explained, it does not provide a basis for reliable conclusions. Therefore, a load test with 300 messages, each separated by a delay of 500 ms, was executed using the process described in 4.2.1.

A visual representation of these messages, as observed in the mobile application, is shown in Figure 4.14.

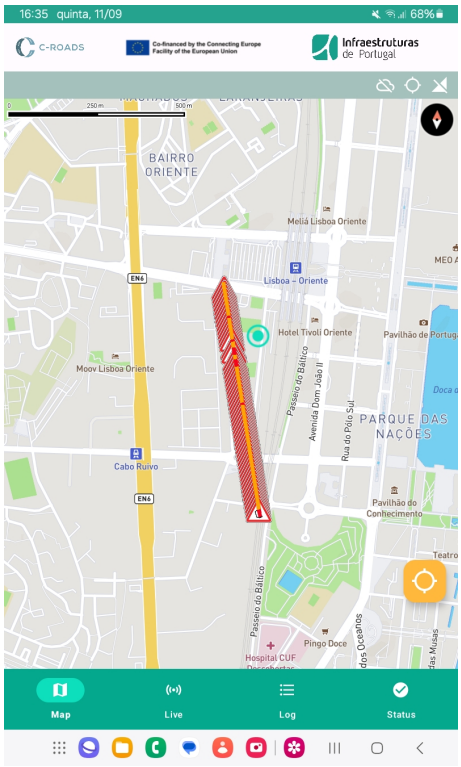


Figure 4.14: 100 Messages in the APP

The average end-to-end time across the 300 messages was 0.402 seconds, which represents a satisfactory result for a pilot architecture. Although still above the ideal target of 100 ms, the result is relatively close. This outcome further reinforces the notion that reliable conclusions cannot be drawn from a single-message measurement,

as the average obtained here differs significantly from the 0.830 seconds measured in the single-message test.

All 300 messages were successfully delivered, demonstrating that the system is capable of withstanding a load of this magnitude.

The relative average delays within and between components are presented in Figure 4.15.

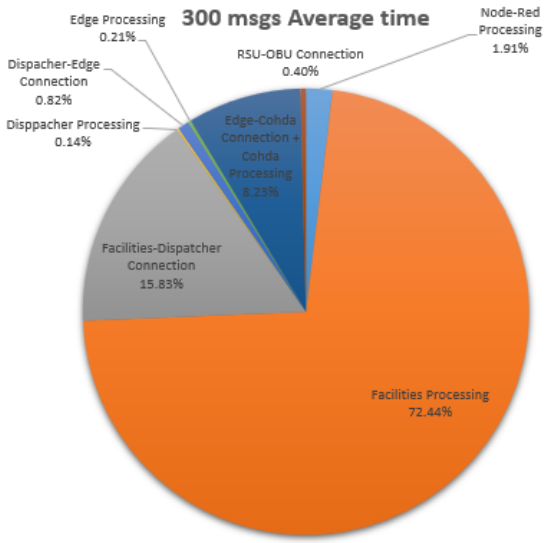


Figure 4.15: Components Delays 300 Message

From the chart, it can be clearly concluded that the component introducing the highest delay is the Facilities Processing, being responsible for more than 70% of it. This result is expected, as it is essentially the only component performing significant processing. Moreover, the Facilities module stores and queries a large volume of information from the database, which is most likely the main source of delay.

A possible solution would be to store all dynamic information currently retrieved from the database in local memory, using technologies such as Redis. This approach would avoid heavy database queries and restrict lookups to the container's own memory, thereby reducing processing time and improving overall system performance. The Facilities component still needs to store the final message in a persistent database, and this operation inevitably introduces some delay. However, this can be mitigated by forwarding the message first and performing the database storage afterward, or even concurrently through separate processes. Such an approach would prevent delays in message dissemination when a high volume of messages needs to be processed.

4.2.3 Delay Tests

This chapter consists of a series of tests in which the number of messages injected into the system remains constant, while the delay between consecutive messages is varied. This approach makes it possible to determine the minimum time interval between messages required for the system to operate as intended.

100 messages with 500ms

First a test with 100 messages with 500ms delay between each message was run, this test is very similar to the one on 4.2.2, and its expected to give similar results.

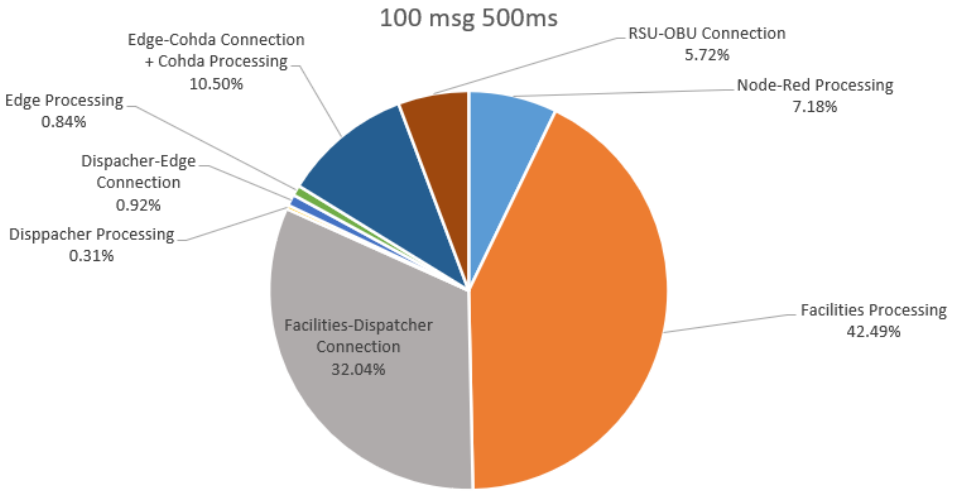


Figure 4.16: 100 Messages with 500 ms Delay

In Figure 4.16, it can be observed that the component introducing the highest delay is Facilities Processing, accounting for 42% of the total time. It is also evident that, when compared with the 300-message load test, the Facilities–Dispatcher connection time has increased substantially. Since these tests were executed on a standard computer, concurrent internal processes likely contributed to the increase in processing times, as all other processing times also showed growth. The overall time end-to-end was 0.570 seconds.

All the messages were delivered without any issue.

100 messages with 250ms

When proceeding to the test regarding 100 messages with 250ms delay time, the results are presented in Figure 4.17.

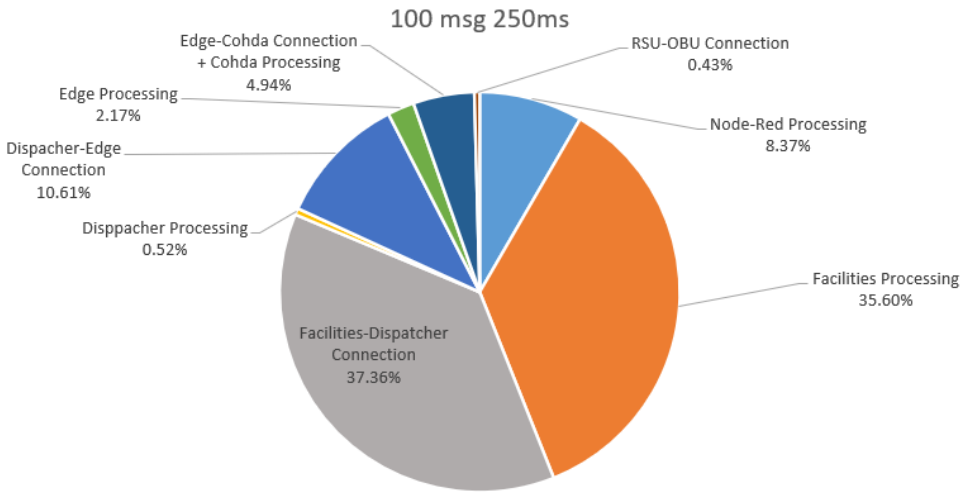


Figure 4.17: 100 Messages with 250 ms Delay

It can be concluded that the overall behavior does not change significantly when compared to the 500 ms delay test. The Facilities–Dispatcher connection, which includes the AMQP, increased by around 5%, but this variation is not considered relevant, as it may result from other parts of the system requiring slightly less time. For instance, the ”Edge–Cohda Connection + Cohda Processing time”, which is highly dependent on network congestion, was reduced by half. The overall time end-to-end was 0.474 seconds.

All the messages were delivered without any issue.

100 messages with 100ms

When analyzing the behavior of the system with 100 messages injected at intervals of only 100 ms, a clear change in performance can be observed. In this scenario, most of the end-to-end delay is concentrated in the AMQP component, which accounts for 92% of the total time as can be seen on Figure 4.18. The overall end-to-end delay also increased considerably, reaching 1.711 seconds.

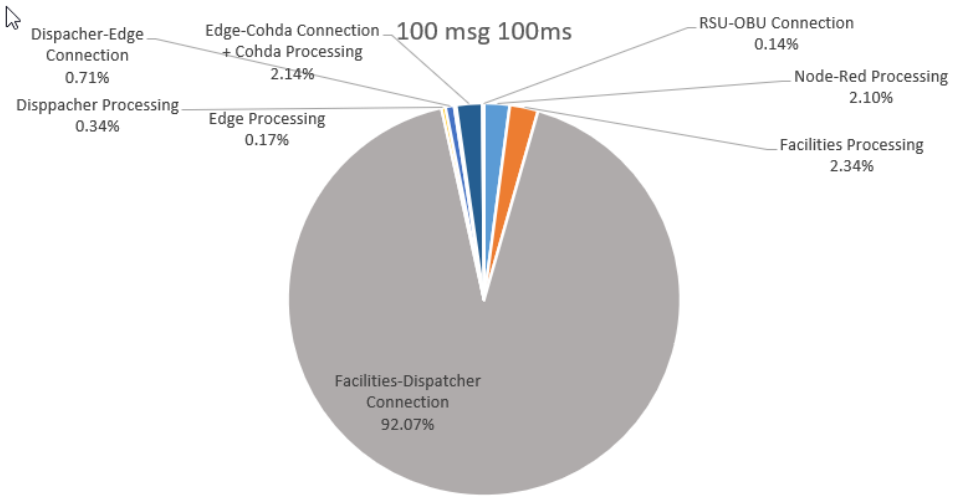


Figure 4.18: 100 Messages with 100 ms Delay

Another important, although not yet critical, observation is the occurrence of message loss. In this case, only a single message was lost during the process. The loss occurred upon arrival at the AMQP component, as the message never progressed beyond that stage. This indicates that, from this point onwards, the system may no longer behave as expected, not only can significant delays in message delivery occur, but issues with successful delivery may also arise.

100 messages with 50ms

When the delay between messages is further reduced to 50 ms, the system begins to collapse, and message delivery is no longer guaranteed. In this test, only 22 out of 100 messages were successfully delivered, with the losses occurring entirely at the AMQP component, and the average end-to-end delay reached 4.571 seconds. From this, it can be concluded that the system's critical threshold lies between 250 ms and 100 ms of inter-message delay. Below this range, the system exhibits unpredictable behavior. As in the previous tests, most of the time is still spent in the AMQP service, as shown in Figure 4.19.

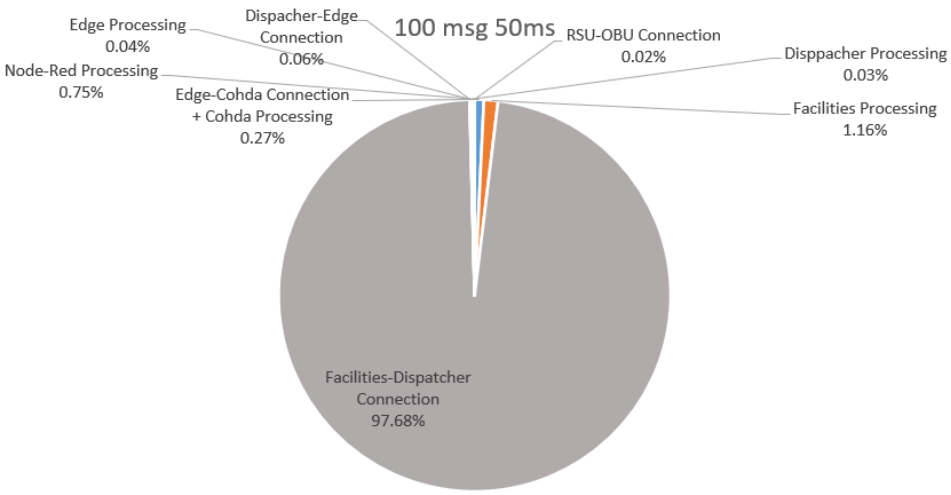


Figure 4.19: 100 Messages with 50 ms Delay

100 messages without Delay

Another important test concerns the system's behavior when a large burst of messages arrives simultaneously. For this purpose, a load test with 100 messages inserted into the system without any delay between them was performed. The average end-to-end delay observed was 17.646 seconds, significantly higher than in the previous tests.

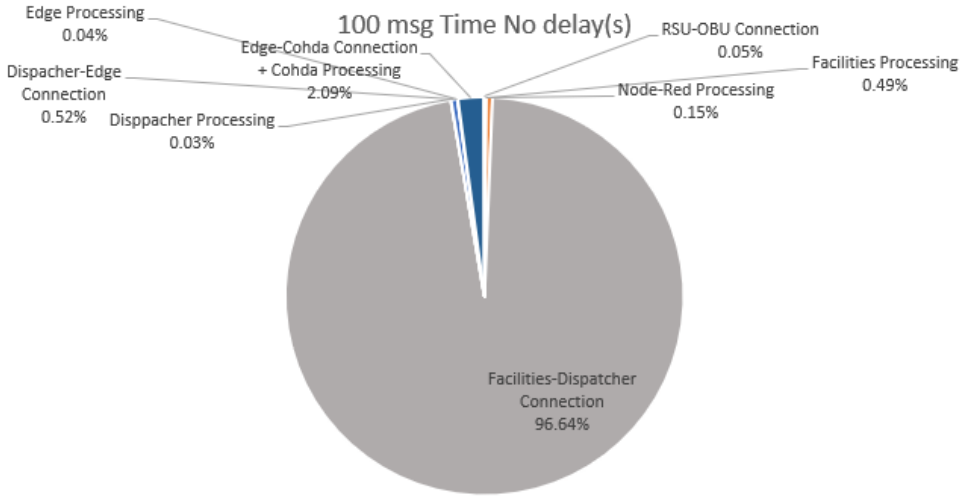


Figure 4.20: Components Delays 100 Message No Delay

In Figure 4.20, it can be observed that when messages are introduced sequentially without delay, the system behaves very differently. An overall slowdown is noticeable, and message delivery is not guaranteed. In this case, only 80 out of 100 messages were successfully delivered. Although this result is considerably better than the 50 ms scenario, it demonstrates that the system behaves inconsistently, making it impossible to predict whether messages will be delivered or not reliably.

A possible reason for this behavior is that, when 100 messages are sent in a

burst without delay to a single consumer, Apache ActiveMQ Artemis applies flow control to protect both the broker and the client from memory overload. The producer is only allowed to transmit while it holds sufficient credits, and the consumer can only buffer messages up to its configured limit. Since the consumer cannot process messages as quickly as they arrive, its buffer becomes saturated, preventing the broker from dispatching additional messages until space is freed. This mechanism explains both the inconsistent throughput and the noticeable delay: the producer pauses while awaiting new credits, and the broker holds back messages until the consumer processes more. The bottleneck, therefore, does not lie in the broker itself but in the consumer's limited processing capacity, which triggers flow control and slows down end-to-end delivery [31].

Assuming this is indeed the root cause, possible solutions include introducing parallelism in the dispatcher, separating processes for message consumption and message forwarding, and increasing the broker's buffer window size to reduce the likelihood of buffer saturation. These measures would address both the message loss issue and the problem of high latency.

Comparison between Delays

In Table 4.1, a comparison is presented between sending messages with different delays, showing the processing times of each component. The results indicate that most components require a similar amount of time to process and forward messages in all scenarios, with the Facilities–Dispatcher connection standing out as the one exhibiting the greatest difference as the delay increases.

Measured Location	500ms	250ms	100ms	50ms	1ms
Node-Red Processing	0.044	0.043	0.037	0.035	0.026
Facilities Processing	0.261	0.184	0.041	0.053	0.087
Facilities–Dispatcher Connection	0.197	0.193	1.609	4.498	17.079
Dispatcher Processing	0.002	0.003	0.006	0.001	0.005
Dispatcher–Edge Connection	0.006	0.055	0.012	0.003	0.092
Edge Processing	0.005	0.011	0.003	0.002	0.006
Edge–Cohda Connection + Processing	0.064	0.026	0.037	0.012	0.369
RSU–OBU Connection	0.035	0.002	0.002	0.001	0.009
Delay End to End	**0.570**	**0.474**	**1.711**	**4.571**	**17.646**

Table 4.1: Measured delays for different message intervals

The results in Table 4.1 clearly show that the system remains stable when messages are sent with intervals of 250 ms or higher, with end-to-end delays below one second. However, as the interval decreases to 100 ms or less, the latency increases sharply, reaching 17.6 s when 100 messages are sent without any delay. The main bottleneck is observed in the Facilities–Dispatcher connection, where the delay grows from only 0.197 s at 500 ms spacing to more than 17 s with no spacing. This indicates that the processing components themselves are not the limiting factor, since their delays remain almost constant, but rather the broker introduces significant flow control overhead when the consumer cannot process messages at the incoming rate. Conse-

quently, the inconsistent throughput and long delays are a direct effect of flow control being triggered to protect memory, highlighting the consumer-side limitations as the critical factor in this scenario

100 Messages with 250ms Hybrid

This test analyzes the end-to-end delay when using the hybrid solution, in which the mobile application is connected to the Edge Cloud broker. The objective is to measure the time taken for a message to travel from Node-RED to the application. Since the mobile application contains logs, it is possible to compare timestamps directly. It should be noted, however, that in this case, time synchronization is not guaranteed, meaning that only an approximate comparison of the efficiency of the hybrid system against the standard C-ITS configuration can be made. A delay of 250 ms was selected, as this value is known to ensure system stability.

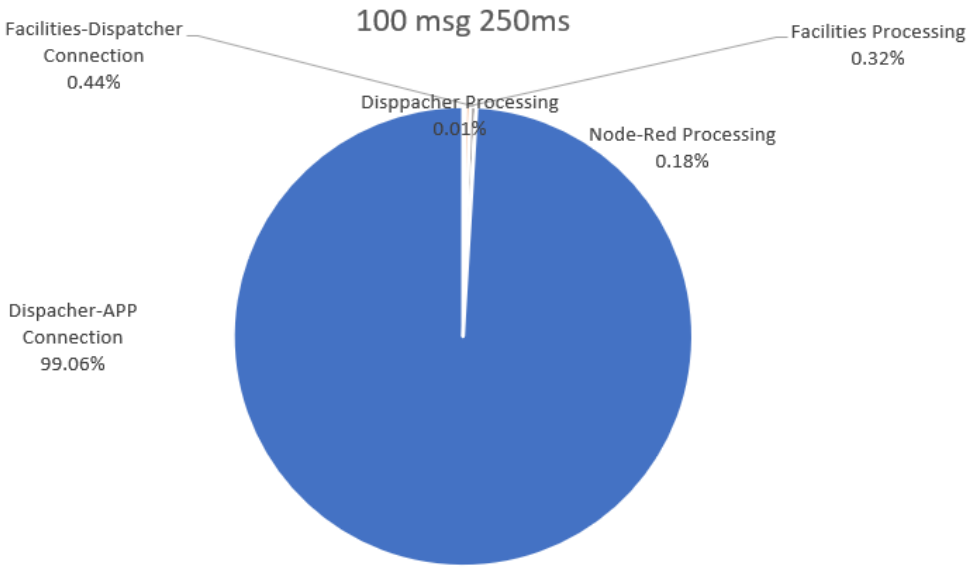


Figure 4.21: Cloud Delay

As shown in Figure 4.21, approximately 99% of the delay occurs along the path from the dispatcher to the application. Although it was not possible to precisely identify the component introducing most of the delay, the Facilities can be excluded as the source in opposition to the results of the previous tests. The average end-to-end delay measured was 47.347 seconds. This reinforces the conclusion that the C-ITS system is significantly more efficient. However, the hybrid system remains usable in certain contexts. For instance, the delivery of roadwork notifications would not be compromised by a delay of up to 50 seconds.

4.3 Fault Tolerance

This chapter evaluates the system's resilience in scenarios where components fail due to malfunction. The analysis focuses on the system's behavior during such

failures, its ability to recover normal operation once the component is restored, and its overall performance under these conditions. The evaluation proceeds from the least likely failures, such as those involving physical components, up to the Central Unit and the databases, which are expected to cause a complete system shutdown. Such testing is only possible in a distributed system where the components operate independently. In this case, since Docker containers are used, the system can continue operating even when an individual component goes down, while still reflecting the malfunction of that specific container.

4.3.1 Physical Components

This subchapter analyzes the deliberate shutdown of the system's physical components, followed by an evaluation of the resulting behavior.

OBU shutdown

When the OBU is turned off, the system continues to operate without issuing any warning, as it has no direct dependency on the OBU devices. The situation is therefore interpreted as a vehicle either leaving the RSU coverage area or being switched off.

The only observable effect is that the mobile application loses access to all information, since it is currently connected to the OBU's MQTT broker. With the OBU disabled, the broker becomes unavailable and no data is transmitted. Under normal operating conditions, the mobile application would instead be connected to the edge server, in which case no noticeable change would occur.

RSU shutdown

When turning off the RSU, the behavior after around 30 seconds can be seen on Figure 4.22

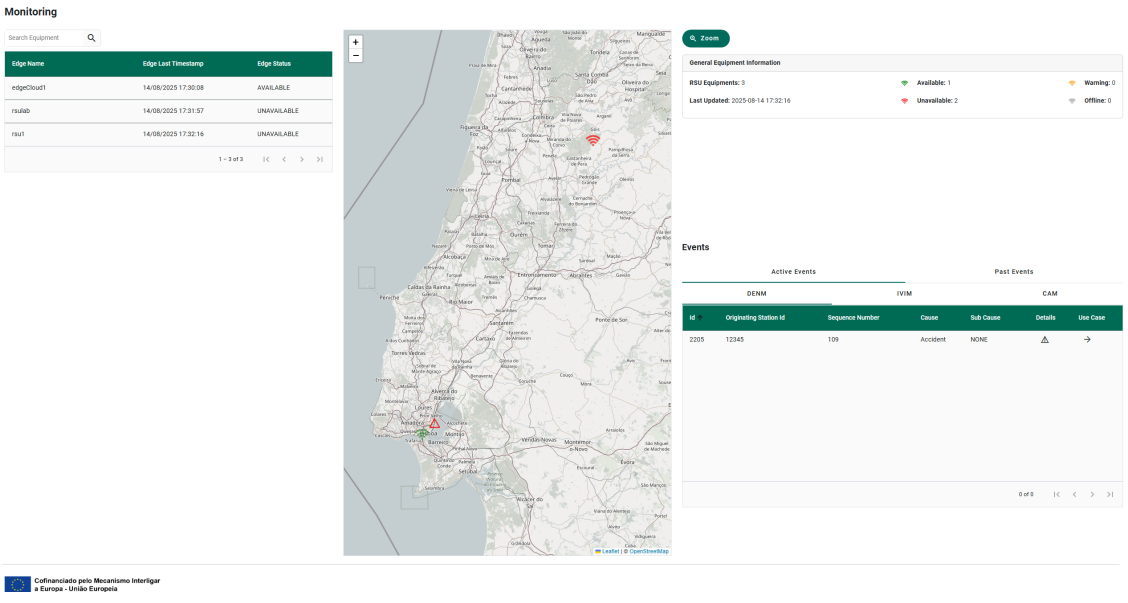


Figure 4.22: RSU Unavailable

Both RSUs appear as “Unavailable” and revert their displayed locations to the default values. No alterations were identified in the system containers. When attempting to generate a message, the application no longer receives C-ITS data, as there is no available source to retrieve it from. Nevertheless, the remainder of the system continues to operate normally, with messages being propagated through the components without interruption. This demonstrates the system’s resilience to RSU faults.

Analysis of the OBU’s RX PCAP file reveals a temporal gap in message reception, corresponding to the moment when the RSU was turned off and the device ceased receiving messages from it. This behavior is evidenced by the temporal discontinuity observed between Figure 4.23 and Figure 4.24.Prior to the RSU shutdown, the OBU was receiving messages from it at a frequency of one message per second.

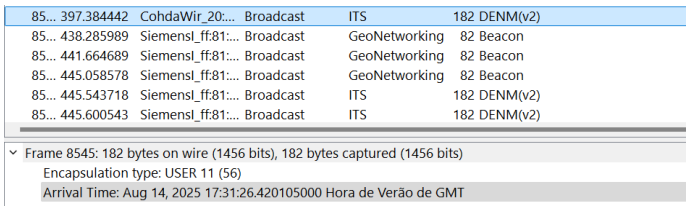


Figure 4.23: RX OBU before RSU shutdown

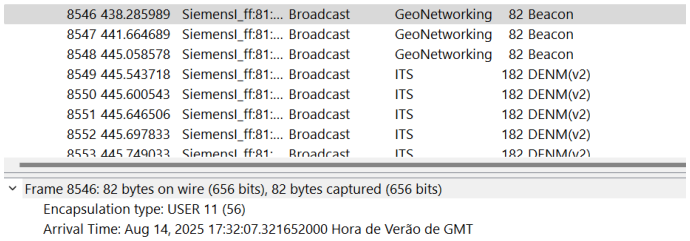


Figure 4.24: RX OBU after RSU reconnection

For approximately 40 seconds, no messages were received, as evidenced by the temporal gap observed between frames 8545 and 8546.

Once the RSUs become available again, the OBU RX PCAP confirms that message reception resumes, but only from the Yunex RSU and not from the Cohda RSU. As shown in Figure 4.24, all subsequent messages are received from “Siemens”(corresponding to the Yunex RSU) and none from “Cohda”. This behavior indicates that the Yunex RSU retains messages in memory and, upon reconnection, verifies their validity. If still valid, the messages are forwarded; otherwise, they are discarded. In contrast, the Cohda RSU does not implement any form of message buffering, resulting in the permanent loss of data once the unit is powered off.

These observations highlight the absence of a resend mechanism in the overall architecture, which would otherwise enable still-valid messages to be periodically resent to the RSUs in the event of outages.

4.3.2 Edge Server

After shutting down the physical components, the next element to be tested to evaluate the system's behavior under failure conditions is the Edge Server.

When all the containers in Edge Server are turned off, no instability is observed in the overall system. However, the MQTT broker detects the loss of connection with both RSUs, Yunex and Siemens as can be seen on Figure 4.25. It can also be noted that, even with the Edge Server unavailable, newly created events still reach the regional node's MQTT broker, but they are not propagated beyond that point, as expected.

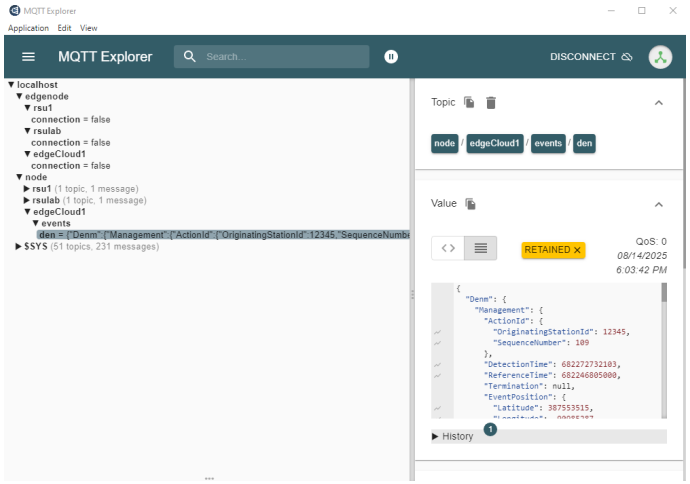


Figure 4.25: MQTT Explorer when Edge Server is Down

When the Edge Server containers are restarted, messages transmitted during the outage are not recovered by the Edge Server. This occurs because the system relies on MQTT, which does not provide a persistent queue. If a consumer is not connected to a producer at the moment a message is published, that message will not be delivered once the consumer reconnects.

A possible solution to address this limitation would be the integration of a Redis database within the dispatcher container at the regional node. Redis is an in-memory database widely used in high-speed, low-latency systems to store temporary data. In this case, still-valid messages would be stored in the Redis database inside the dispatcher container, and upon reconnection, they would be forwarded to the Edge Server. This approach would ensure that every Edge Server joining or reconnecting to the system has access to all relevant messages being transmitted within that region.

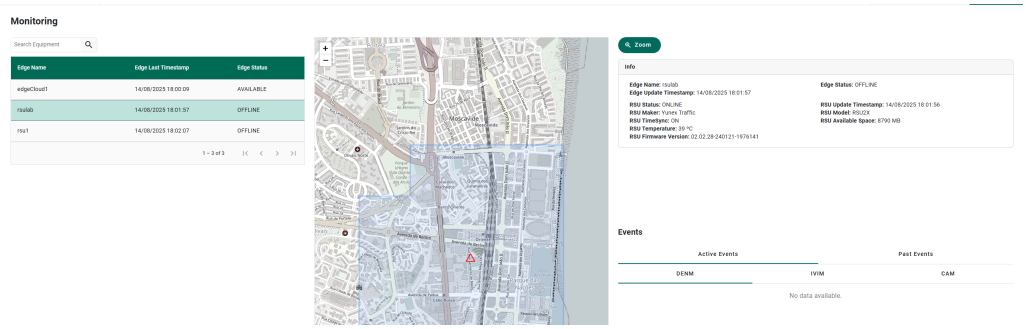


Figure 4.26: GUI when Edge Server is Down

In the portal, the status of the Edge Servers is displayed as offline (Figure 4.26), indicating that they are no longer operational. The last known state of the physical devices is shown. In this case, when the Edge Server went down, the devices were online. The last known location of the RSU remains available rather than reverting to a default value, which represents a desirable behavior.

As expected, if an RSU fails under these conditions, its status is not updated, since there is no direct connection between the central system and the physical devices.

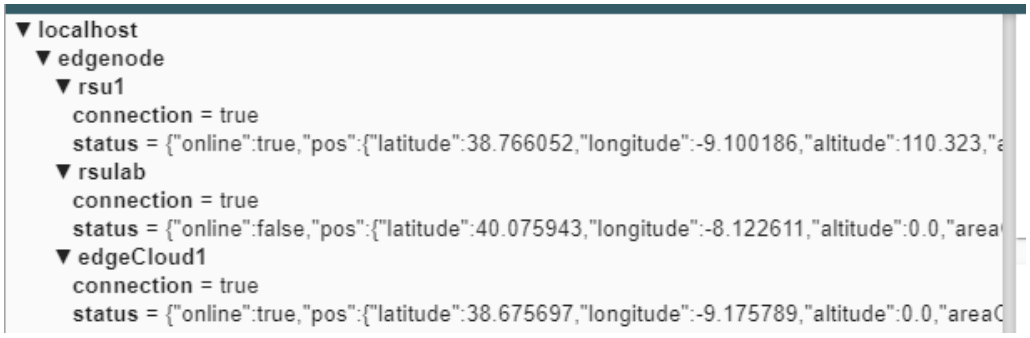


Figure 4.27: MQTT Explorer when Edge Server is UP

When the Edge Server is restarted, the connection is re-established almost instantly, and both the current location and the status of the RSUs become immediately available, as can be seen on Figure 4.27. From that moment, it is possible to resume message transmission without any issues.

Containers Inside Edge Server Shutdown Individually

When the Yunex Edge, Edge Cloud, and Cohda Edge are individually shut down, system stability is preserved, with only the information in the MQTT broker being updated (Figure 4.28) and consequently reflected in the portal.

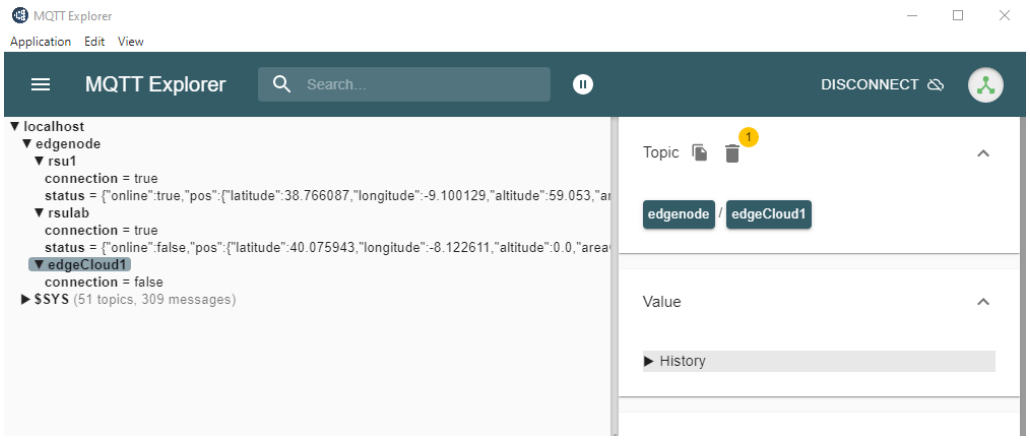


Figure 4.28: MQTT Explorer when Edge Cloud is Down

When the MQTT container within the Edge Server is shut down, both the Edge Cloud and the Yunex Edge continuously attempt to reconnect, but no restart or instability occurs. This is the expected and correct behavior.

In the Cloud container interface, messages such as “MQTTBroker Offline!” and “MQTT Status Sent!” are displayed, and this information is propagated to the portal, where the “RSU status” appears as offline (Figure 4.29). Since no physical equipment exists in the cloud, this is, in fact, the correct place to display such information.

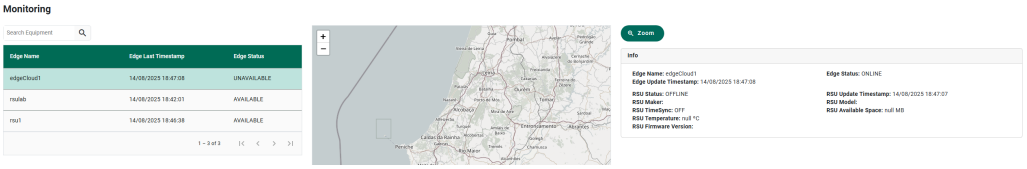


Figure 4.29: RSU Status when MQTT Broker Down

4.3.3 Shutdown Regional Node

The panel displayed in Figure 4.30 shows information that is not entirely realistic, since the central server has no means of directly monitoring the status of the edges or the physical devices, displaying only the last known state. Tests performed by shutting down both the edges and the physical devices confirmed that no changes were reflected in their status indicators.

A more accurate approach for the system would be to remove a regional node and all associated information as soon as the node goes down. This would prevent obsolete nodes from remaining visible and avoid inconsistencies in message status. Furthermore, each time a new regional node registers itself with the AMQP broker within the Central system, all still-valid messages within its geographical tiles should be retransmitted, ensuring that the system state remains fully up to date. This information regarding still-valid messages, similar to what has been proposed for the Edge Server, should be stored in a dynamic database, such as Redis.

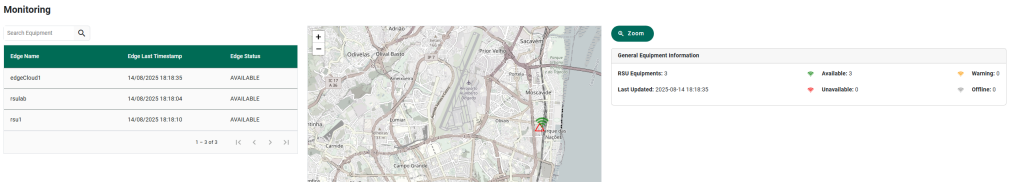


Figure 4.30: GUI when Regional Node is Down

Looking back at the system, the RSU containers within the Edge Server continuously attempt to reconnect to the MQTT broker of the Regional Node. This represents the expected and correct behavior of the components.

When the Node is restarted, system stability is restored, and it becomes immediately possible to resume message transmission, with the OBU receiving the data without issues.

When only the MQTT service within the Node is shut down, the dispatcher continuously attempts to reconnect but remains stable, without crashing.

4.3.4 Shutdown Control Unit

When the entire central system is shut down, no immediate changes occur in the remaining components. The dispatcher continuously attempts to reconnect to the AMQP broker but remains stable and does not crash. However, as expected, the overall service becomes completely unavailable. Once the system is restarted, service is restored within a few seconds.

When the Facilities component is shut down, no immediate reaction is observed in the system. However, when a message is sent, an error occurs in the Node-RED container, as expected, with the following description: “RequestError: getaddrinfo ENOTFOUND cits-svc-facilities”.

When the AMQP broker is shut down, the dispatcher continuously attempts to reconnect but remains stable and does not terminate. The Facilities component exhibits the same behavior.

When the central system databases are shut down, all information regarding the edges and physical devices is lost, and the portal displays “no data available” across all pages, as can be seen on Figure 4.31. This is the expected Behavior, as the Containers are not supposed to work without a DB.

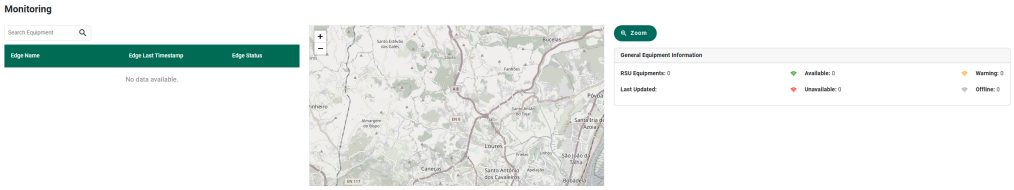


Figure 4.31: GUI when DB is Down

5
Conclusions and Future Work

This thesis analyzed the integration of C-ITS with legacy vehicles through a hybrid architecture based on MQTT. The work addressed a central limitation of current deployments, which mainly benefit vehicles equipped with On-Board Units, and proposed an inclusive solution capable of extending real-time information dissemination to mobile devices.

The experimental evaluation of *Infraestruturas de Portugal*'s industrial architecture demonstrated the technical feasibility of thir architecture. Under controlled loads of up to 300 messages with 500 ms spacing, the system maintained an average latency of approximately 0.4 seconds, a satisfactory result for a pilot-stage prototype. Nonetheless, significant bottlenecks were identified, particularly in the *Facilities* module and in the *Facilities–Dispatcher* connection via AMQP. These components showed limited scalability when the inter-message interval fell below 250 ms, highlighting the system's current constraints. Furthermore, resilience tests confirmed that the architecture preserves stability in the event of partial failures, although it lacks some mechanisms for message retransmission, resulting in loss of data during outages.

Beyond its technical contributions, this work also delivered practical value by reinforcing IP's C-ITS architecture and supporting its ongoing development and validation. From an academic perspective, the research provides a foundation for further exploration of hybrid communication models that combine ITS-G5 and MQTT, as well as the adoption of in-memory databases to minimize processing delays and enhance availability.

Future work should focus on three main directions:

- **Optimization of core modules:** improving the performance of the *Facilities* service through parallel processing and temporary in-memory storage, while decoupling message forwarding from database persistence.

- **Resilience enhancement:** introducing robust retransmission and buffering mechanisms to ensure that valid messages are not lost during component failures or disconnections.

- **Real-world validation:** extending tests beyond laboratory conditions by involving diverse user profiles (drivers, pedestrians, cyclists) and deploying the system

in large-scale field trials, to validate its capacity for real-time and mission-critical use cases.

In conclusion, this thesis demonstrated the technical feasibility of building an effective bridge between modern and legacy vehicles, thereby contributing to the uniformization of traffic information access. By enabling wider participation in the C-ITS ecosystem, the proposed hybrid solution enhances road safety, promotes co-operative mobility, and lays the groundwork for scalable, inclusive, and future-ready intelligent transportation systems. Furthermore, it provides a viable reference architecture that can serve as a foundation for future developments and market adoption. It is important to emphasize, however, that the tested architecture still requires significant improvements to achieve production-level robustness, although it represents an excellent basis upon which a more resilient and reliable system can be built in the future.

Bibliography

[1] Eurostat, "Road safety statistics in the eu," 2023, accessed: 18 September 2025. [Online]. Available: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Road_safety_statistics_in_the_EU

[2] L. J. A. Goncalves, "An ecosystem for securing vehicle-to-everything communication," Masters Thesis, Instituto Superior Técnico, Lisbon, Portugal, 2019. [Online]. Available: https://fenix.tecnico.ulisboa.pt/downloadFile/1689244997260084/86319-leonardo-goncalves_resumo.pdf

[3] N. Mellegård and F. Reichenberg, "The day 1 c-its application green light optimal speed advisory—a mapping study," *Transportation Research Procedia*, vol. 49, pp. 170–182, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352146520307365

[4] C-Roads Platform, "C-roads platform - objectives," 2025, accessed: 10 March 2025. [Online]. Available: https://www.c-roads.eu/platform/objectives.html

[5] C.-R. Platform, "The c-roads platform – an overview of harmonised c-its deployment in europe," *C-Roads Journal*, 2021. [Online]. Available: https://www.c-roads.eu/fileadmin/user_upload/media/Dokumente/C-Roads_Brochure_2021_final_2.pdf

[6] J. Kang, S. Tak, and S. Park, "Analyzing the impact of c-its services on driving behavior: A case study of the daejeon–sejong c-its pilot project in south korea," *Sustainability*, vol. 15, no. 16, p. 12655, 2023. [Online]. Available: https://www.mdpi.com/2071-1050/15/16/12655

[7] Infraestruturas de Portugal, "C-streets project – cooperative intelligent transport systems (c-its)," 2023, accessed: 10 March 2025. [Online]. Available: https://www.infraestruturasdeportugal.pt/pt-pt/node/377

[8] CAR 2 CAR Communication Consortium, "About c-its – car 2 car communication consortium," 2025, accessed: 10 March 2025. [Online]. Available: https://www.car-2-car.org/about-c-its

[9] S. Ravidas, P. Karkhanis, Y. Dajsuren, and N. Zannone, "An authorization framework for cooperative intelligent transport systems," in *Lecture Notes in Computer Science*. Springer, 2020. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-39749-4_2

[10] European Commission, “European its committee – 28th meeting, c-its update,” European Commission, DG JRC E.3: Cyber and Digital Citizens’ Security, Brussels, Belgium, Tech. Rep., May 2020, presentation on C-ITS developments, European Certificate Trust List (ECTL), and EU Root CA. [Online]. Available: https://ec.europa.eu/transport/themes/its/c-its_en

[11] E. C. J. R. C. (JRC), “C-its point of contact (cpoc) protocol,” European Commission, Tech. Rep. Release 1.2, December 2021, accessed: 10 March 2025. [Online]. Available: https://cpoc.jrc.ec.europa.eu/Documentation.html

[12] S.-C. Arseni, D. Avram, M. Medvei, M. Togan, and A. Dima, “Securing the c-its: A pki perspective,” *Romanian Cyber Security Journal*, vol. 3, no. 1, pp. 13–25, 2021. [Online]. Available: https://rocys.ici.ro/spring-2021-no-1-vol-3/securing-the-c-its-a-pki-perspective/

[13] ETSI, “Intelligent transport systems (its); security; trust and privacy management,” European Telecommunications Standards Institute (ETSI), Tech. Rep. TS 102 941 V1.3.1, February 2019, accessed: 10 March 2025. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/102900_102999/102941/01.03.01_60/ts_102941v010301p.pdf

[14] ——, “Intelligent transport systems (its); vehicular communications; basic set of applications; part 2: Specification of cooperative awareness basic service,” European Telecommunications Standards Institute (ETSI), Tech. Rep. EN 302 637-2 V1.3.1, September 2014, accessed: 11 March 2025. [Online]. Available: https://www.etsi.org/deliver/etsi_en/302600_302699/30263702/01.03.01_30/en_30263702v010301v.pdf

[15] ——, “Intelligent transport systems (its); users and applications requirements; part 2: Applications and facilities layer common data dictionary; release 2,” European Telecommunications Standards Institute (ETSI), Tech. Rep. TS 102 894-2 V2.1.1, November 2022, accessed: 11 March 2025. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/102800_102899/10289402/02.01.01_60/ts_10289402v020101p.pdf

[16] ——, “Intelligent transport systems (its); vehicular communications; basic set of applications; part 3: Specifications of decentralized environmental notification basic service,” European Telecommunications Standards Institute (ETSI), Tech. Rep. EN 302 637-3 V1.3.1, April 2019, accessed: 11 March 2025. [Online]. Available: https://www.etsi.org/deliver/etsi_en/302600_302699/30263703/01.03.01_60/en_30263703v010301p.pdf

[17] ——, “Intelligent transport system (its); vehicular communications; basic set of applications; collective perception service; release 2,” European Telecommunications Standards Institute (ETSI), Tech. Rep. TS 103 324 V2.1.1, June 2023, accessed: 15 March 2025. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/103300_103399/103324/02.01.01_60/ts_103324v020101p.pdf

[18] ——, "Intelligent transport systems (its); vulnerable road users (vru) awareness; part 3: Specification of vru awareness basic service; release 2," European Telecommunications Standards Institute (ETSI), Tech. Rep. TS 103 300-3 V2.1.1, November 2020, accessed: 15 March 2025. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/103300_103399/10330003/02.01.01_60/ts_10330003v020101p.pdf

[19] ——, "Intelligent transport systems (its); vehicular communications; basic set of applications; facilities layer protocols and communication requirements for infrastructure services," European Telecommunications Standards Institute (ETSI), Tech. Rep. TS 103 301 V1.3.1, February 2020, accessed: 15 March 2025. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/103300_103399/103301/01.03.01_60/ts_103301v010301p.pdf

[20] B. J. M. Silva, "Plataforma de gestão de serviços c-its," Master's Thesis, Universidade de Aveiro, Aveiro, Portugal, 2024, disponível online: https://doi.org/10.54499/UIDB/50008/2020, Acedido em: 15 Março 2025.

[21] IP Telecom - Portugal, "Ccms - c-its credential management system," 2025, accessed: 10 March 2025. [Online]. Available: https://github.com/iptelecom-portugal/ccms.pt

[22] A. Hugo, B. Morin, and K. Svantorp, "Bridging mqtt and kafka to support c-its: A feasibility study," in *21st IEEE International Conference on Mobile Data Management (MDM)*. IEEE, 2020, pp. 371–376. [Online]. Available: https://ieeexplore.ieee.org/document/9162327

[23] E. Vieira, J. Almeida, J. Ferreira, T. Dias, A. Vieira Silva, and L. Moura, "A roadside and cloud-based vehicular communications framework for the provision of c-its services," *Information*, vol. 14, no. 3, p. 153, 2023. [Online]. Available: https://www.mdpi.com/2078-2489/14/3/153

[24] M. J. Jooriah, J. A. Almeida, and J. Ferreira, "Hybrid routing mechanism for wireless vehicular networks," in *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*. IEEE, 2021, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/9448709

[25] N. R. P. Lopes, "A secure data dissemination scheme for connected vehicle networks," Master's thesis, Instituto Superior Técnico, Universidade de Lisboa, Lisbon, Portugal, Jun. 2024.

[26] J. Ferreira, J. Fonseca, D. Gomes, J. Barraca, B. Fernandes, J. Rufino, J. Almeida, and R. Aguiar, "Pasmo: An open living lab for cooperative its and smart region," in *2017 International Smart Cities Conference (ISC2)*. IEEE, 2017, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/8090866

[27] C.-M. P. Team, "D3.3 low-level implementation-ready architecture," *C-MobILE*, 2025. [Online]. Available: https://www.c-mobile-project.eu

[28] Infraestruturas de Portugal, SA, “Implementação de infraestrutura de suporte aos serviços c-its e ccam na rede de alta prestação (rap) sob gestão direta da ip – desenvolvimento plataforma c-its e app – c-streets,” Infraestruturas de Portugal, Direção de Acessibilidade, Telemática e ITS – Departamento ITS e Acessibilidade, Almada, Portugal, Tech. Rep., January 2022, caderno de Encargos - Condições Técnicas, Processo DESCO nº 10008891. [Online]. Available: https://my.alertaconcursospublicos.pt/concursos/9379

[29] eBay, “Road side unit (rsu) - cohda wireless mk5,” https://www.ebay.com/itm/114784346299, n.d., accessed: 2025-07-27.

[30] Y. Traffic, “First connected vehicle roadside unit proven to talk to automakers’ 2023 models,” https://us.yunextraffic.com/newsroom/first-connected-vehicle-roadside-unit-proven-to-talk-to-automakers-2023-models/, 2023, accessed: 2025-07-27.

[31] Apache ActiveMQ Artemis Documentation, “Flow control,” 2017, accessed: 10 March 2025. [Online]. Available: https://activemq.apache.org/components/artemis/documentation/2.3.0/flow-control.html