ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN
COMPUTER SCIENCE AND ENGINEERING
Ciclo 37

Settore Concorsuale: 09/H1 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI
Settore Scientifico Disciplinare: ING-INF/05 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

VEHICULAR-BASED SUPPORT TO CUTTING-EDGE APPLICATION SCENARIOS IN NEXT-GEN NETWORKS

Presentata da: Angelo Feraudo

Coordinatore Dottorato:
Prof.ssa Ilaria Bartolini

Supervisore:
Prof. Paolo Bellavista

Esame finale anno 2025

Abstract

The rapid advancements in in-vehicle computing, communication, and software enable drivers to access diverse distributed applications and services. Edge Computing and frameworks like the European Telecommunications Standards Institute (ETSI) Multi-access Edge Computing (MEC) are intended to be key in standardizing service execution at the network edge. Complementing this, Vehicular Cloud Computing (VCC) and Vehicular Edge Computing (VEC) augment edge computational capacity by harnessing vehicle computing, storage, and communication resources. Together, these technologies create a resource continuum, enabling real-time processing, low latency, and improved service quality. This dissertation presents several solutions to contribute to the creation of an innovative and powerful vehicular-based support for edge applications in next-generation networks. Hence, it extensively studies and contributes to enhancing vehicular communication reliability and advancing the vehicular computing paradigm by developing simulated/emulated platforms for application design in this domain. The key contributions of this PhD dissertation include: i) a reputation-based system to exclude misbehaving vehicles within Vehicle-to-Vehicle (V2V) communications, ii) a dataset comprising V2V messages compliant with European standards, iii) an extensive study on the usage of a Time-Sensitive Networking (TSN) middleware to improve Quality of Service (QoS) and data delivery in Vehicle-to-Everything (V2X) communications, iv) the design of a vehicular computing architecture based on recognized standards (e.g., MEC) to promote interoperability and scalability, and v) a novel simulation tool allowing researchers and engineers to design, test, and enhance distributed applications that exploit vehicular, edge, and cloud computing concepts.

Furthermore, this dissertation explores MEC deployment within 5G networks using Open Radio Access Network (O-RAN) specifications, establishing a foundation for a real-world vehicular cloud computing testbed. This led to an openly accessible middleware for developing near real-time applications following O-RAN standards, essential for integrating MEC and O-RAN. All proposed solutions have been rigorously tested, demonstrating their effectiveness in advancing next-generation vehicular and edge computing.

To my family

Contents

List of Figures

List of Tables

Acronyms

3GPP	3rd Generation Partnership Project
AMF	Access and Mobility Management Function
AMS	Application Mobility Service
AoI	Area of Interest
BSM	Basic Safety Message
BTP	Basic Transfer Protocol
C-ITS	Cooperative-Intelligent Transportation System
C-V2X	Cellular-Vehicle-to-everything
CAM	Cooperative Awareness Message
CAN	Controller Area Network
CFS	Customer Facing Service
COTS	Commercial Off-the-Shelf
CSMA/CA	Carrier-Sensing Multiple Access Collision Avoidance
CU	Central Unit
DAG	Direct Acyclic Graph
DENM	Decentralized Environmental Notification Message
DHCP	Dynamic Host Configuration Protocol
DID	Decentralized Identifier
DIVA	Decentralized Identifiers (DID)-based reputation system for secure transmission in VAnets
DLT	Distributed Ledger Technology
DU	Distributed Unit
E2AP	E2 Application Protocol
E2SM	E2 Service Model
ECU	Electronic Control Unit
ETSI	European Telecommunications Standards Institute
FSM	Finite State Machine
gNB	Next Generation NodeB

GPS	Global Positioning System
ITS	Intelligent Transportation System
KPM	Key Performance Measurement
MAC	Medium Access Control
MEC	Multi-access Edge Computing
MEC App	MEC Application
MEC-H	MEC-Host
MEC-O	MEC Orchestrator
MEC-P	MEC Platform
MEC-PM	MEC Platform Manager
NR	New Radio
O-RAN	Open Radio Access Network
OAI	Open Air Interface
OBU	Onboard Unit
OMNeT++	Objective Modular Network Testbed in C++
OSC	O-RAN Software Community
OSI	Open Systems Interconnection
OTA	Over-The-Air
PTP	Precision Time Protocol
QoS	Quality of Service
RAN	Radio Access Network
RC	RAN Control
RIC	RAN Intelligent Controller
RMR	RIC Message Router
RNI	Radio Network Information
RRM	Radio Resource Management
RSU	Road Side Unit
RT	Real-Time
RU	Radio Unit
SDL	Shared Data Layer
SINR	Signal to Interference Noise Radio
SM	Service Model
SMF	Session Management Function
SMO	Service Management and Orchestration
TA	Trusted Authority

TSN	Time-Sensitive Networking
UALCMP	User Application LifeCycle Management Proxy
UAV	Unmanned Aerial Vehicle
UE	User Equipment
UPF	User Plan Function
URI	Uniform Resource Identifier
V2I	Vehicle-to-infrastructure
V2V	Vehicle-to-vehicle
V2X	Vehicle-to-everything
VANET	Vehicular ad hoc network
VC	Vehicular Computing
VCC	Vehicular Cloud Computing
VCred	Verifiable Credential
VEC	Vehicular Edge Computing
VFC	Vehicular Fog Computing
VI	Virtualization Infrastructure
VIM	Virtualization Infrastructure Manager
VP	Verifiable Presentation
WAVE	Wireless Access in Vehicular Environments

1 Introduction

The rapid advancement of computing technologies and next-generation wireless communication networks has revolutionized in-vehicle hardware, turning vehicles into powerful mobile computation nodes on wheels [1, 2, 3, 4]. New-gen vehicles include Onboard Unit (OBU)s and onboard radio modules, such as IEEE 802.11p [5] or 5G modules [6], enabling environmental perception, computational processing, and communication capabilities. As a result, they now play a central role in Intelligent Transportation Systems (ITSs), which aim to enhance the efficiency and safety of transportation networks through advanced data exchange and communication technologies. In this scenario, drivers can tap into an extensive array of distributed applications and services, often cloud-based, as in other vertical domains, including improved access to information and entertainment features. This creates a new ecosystem with unique real-time demands, presenting substantial challenges for the backbone network and cloud infrastructure [7].

Vehicular ad hoc network (VANET) play a key role in such an environment, as they enable the communication between vehicles (V2V) and infrastructure (V2I) equipment, providing drivers and traffic authorities with information related to traffic conditions and incidents [8]. Similarly, edge computing has emerged as a facilitator for this new ecosystem by providing computational resources closer to the end-users [9, 10]. It extends traditional cloud infrastructure by deploying resources at the network edge, near data sources, thereby offering a more efficient and scalable approach to meet the growing demand for computing and storage. To accelerate the adoption of this paradigm, the European Telecommunications Standards Institute (ETSI) has proposed the Multi-access Edge Computing (MEC) standard [11]. This standard enables running contextualized MEC-compliant applications near the data sources and/or users and within a virtualized and multi-tenant environment. Furthermore, the MEC standard facilitates the integration of cloud resources with those available at the edge, creating a complete cloud continuum of virtualized resources distributed within the network.

As more businesses begin to leverage the shared edge-cloud environment, new opportunities and challenges emerge. In contrast to traditional cloud deployments in data centers, the edge has limited resources and may not always be able to satisfy application demands for resources and associated Quality of Service (QoS). Moreover, it is predicted that by 2025, a single connected vehicle

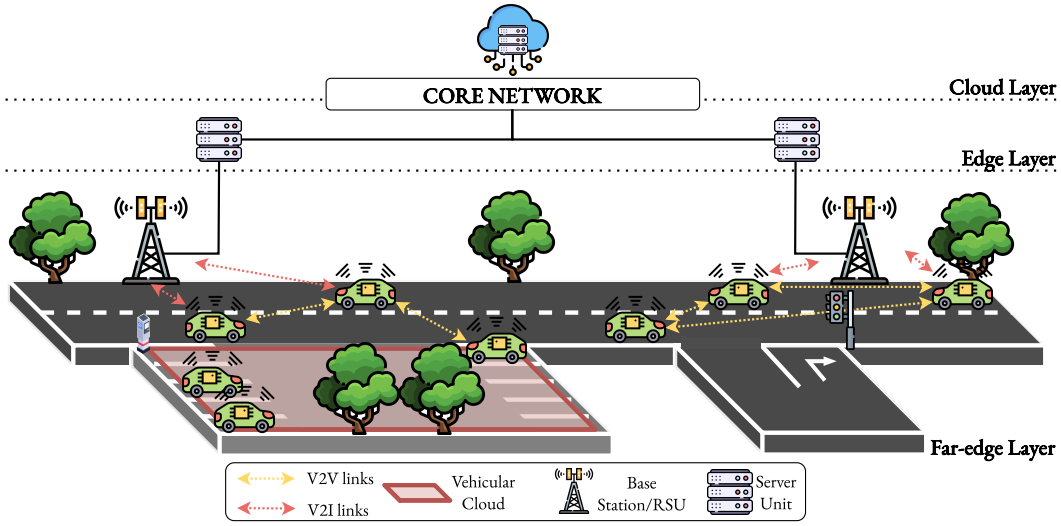


Figure 1.1: Vehicular Computing Layers Structure

could generate between 1 to 10 exabytes of data traffic per month for cloud-based services [12], placing significant strain on network infrastructures. For these reasons, there is a pressing need to dynamically identify and integrate additional resources to support edge and network infrastructure, thereby ensuring service availability in densely populated and congested deployment scenarios. Vehicular Cloud Computing (VCC) has emerged as a promising paradigm to support distributed applications designed according to the edge cloud approach [2, 13, 14, 15]. VCC leverages both cloud resources and the computing, storage, and communication capabilities available in vehicles to create cost-effective mobile clouds at the very edge of the network. This paradigm expects vehicles to autonomously form vehicular clouds,i.e., aggregation of vehicular resources, through VANET. Complementing VCC, Vehicular Edge Computing (VEC) [7, 3] introduces edge computing in providing services and resources to vehicles. In this scenario, Road Side Unit (RSU) are considered off-grid servers (edge nodes) placed close to vehicles, thereby minimizing response time and the amount of data transmitted on the network.

Given that these paradigms - VCC and VEC - often overlap and may cause confusion, we use the term Vehicular Computing (VC) in this dissertation to refer to an integrated paradigm that encompasses resources available at the cloud, edge, and “far-edge" to support vehicular applications and services. Additionally, since fog computing is a generalized form of edge computing [16], VC also incorporates the Vehicular Fog Computing (VFC) paradigm. The layered architecture that defines this concept is illustrated in Figure 1.1, which synthesizes various existing models [7, 17, 3] into a unified approach.

To enable the VC paradigm and, thus, to support the deployment of software components across different vehicle computer variants, the traditional embedded platform of vehicles needs to be replaced with a software-defined architecture [2]. This architecture allows vehicles to support cloud-native technologies and receive Over-The-Air (OTA) updates throughout their life cycle. A notable effort in this direction is the ARM's Scalable Open Architecture for Embedded Edge (SOAFEE) project [18], which offers an open-source reference implementation enabling a cloud-native architecture hardware agnostic for the vehicular scenario.

This dissertation investigates and proposes novel technologies to support this new ecosystem, which involves deploying next-generation edge applications within the vehicular context. It focuses mainly on two areas: vehicular networking, as this system enables the execution of services in close proximity to the requester, and vehicular computing, to leverage the computational power of vehicles and improve service availability and resource efficiency in dynamic environments.

Chapter 2 provides essential background on the elements relevant to this dissertation. It begins with an introduction to vehicular networking concepts and the European standards governing this technology. Next, it presents an overview of the 3rd Generation Partnership Project (3GPP) standards that define the 5G Radio Access Network (RAN) and Core. The chapter then covers the ETSI MEC standard, explaining its components and integration within a 5G network. Additionally, it introduces the VC paradigm, offering an overview of the potential layering model in this context. Finally, it discusses key design aspects of the Open Radio Access Network (O-RAN) specification, focusing on the elements relevant to this dissertation.

Chapter 3 investigates innovative solutions to enhance the reliability and integrity of vehicular communications, with a particular focus on standard message formats such as Cooperative Awareness Message (CAM) and Decentralized Environmental Notification Message (DENM) [19, 20]. It introduces a robust solution for identifying malicious messages within vehicular networks using a reputation-based mechanism. Additionally, it proposes an approach that improves data delivery in car platooning (i.e., cooperation among a group of vehicles) scenarios using Vehicle-to-vehicle (V2V) communications, optimizing message exchange for real-time traffic coordination and safety.

Chapter 4 explores the design of an architecture that leverages ETSI MEC to enhance edge node computational capacity by exploiting underutilized vehicular resources. Thus, this chapter focuses on creating a vehicular computing paradigm that relies on well-established standards, improving service availability within highly mobile networks, and offering a flexible and scalable solution for resource-constrained environments. It also details the implementation of a simulation tool designed to model and test next-generation vehicular applications, employing the concept of VC to create a distributed, scalable network of mobile edge resources.

Chapter 5 conducts an in-depth analysis of integrating edge computing into cellular networks, particularly in the context of 5G. It explores the synergy between ETSI MEC and O-RAN to enable the deployment of MEC-compliant applications in 5G environments. This research lays the foundation for the creation of a real-world testbed for the VC paradigm, building upon the architecture proposed in Chapter 4. Furthermore, it outlines the creation of a framework for developing near-real-time applications dedicated to RAN monitoring and management, offering significant improvements in the development process of these applications.

Chapter 6 summarizes the key conclusions of this dissertation, highlighting its contributions and the advancements made in the field. It also explores promising future research directions based on the findings, suggesting areas for further development and potential applications of the proposed approaches.

This dissertation provides both theoretical advancements and practical solutions for enhancing service continuity in the evolving landscape of vehicular edge applications, bridging the gap between vehicular networking, computing, and next-generation cellular technologies.

1.1 Challenges and Contributions overview

ETSI has made significant advancements in the field of ITS with the publication of several documents aiming to constitute a set of standards for the development of Cooperative-Intelligent Transportation System (C-ITS) [21]. It defines message structures for both periodically exchanged messages (non-safety messages) [19] and messages exchanged in exceptional situations (safety messages) [20]. Ensuring the trustworthiness of these messages is crucial, as they play a central role in maintaining traffic safety. Moreover, the loss or tampering of such messages can disrupt the correctness of some critical functionality, such as autonomous driving, by significantly increasing the risk of accidents.

Similarly, VANET scenarios present a multitude of new challenges [22, 23, 24, 25], especially regarding security and reliability. From a security perspective, attackers may exploit vulnerabilities to forge or alter messages, propagating false or misleading information through the network. This can result in potentially severe consequences, such as traffic accidents or misrouted vehicles. On the data delivery side, the transmission of cooperative awareness messages usually occurs at a given frequency, thus exacerbating the probability of data loss and leading to repeated interference between nodes [25]. Addressing these challenges requires developing solutions that ensure both the authenticity and integrity of messages while mitigating interference and preventing data loss.

Furthermore, as vehicles evolve into both data sources and mobile computing nodes, the demand for computational resources near the edge of the network grows. This necessitates the effi-

cient allocation and utilization of resources to ensure timely and reliable service delivery. Several frameworks and models [2, 4] proposed to incorporate/exploit resources included in software-defined vehicles into/with the cloud infrastructure, ensuring seamless connectivity and resource utilization. Regarding task allocation, various algorithmic strategies have been suggested to assess the likelihood of nodes successfully completing computational tasks [26, 27, 28].

Nevertheless, many existing solutions ignore the challenges of multi-vendor and multi-domain environments, where interoperability and coordination between different systems are critical. Additionally, the issue of task migration, transferring ongoing tasks between vehicles or edge nodes to maintain service continuity, remains underexplored. Finally, there is a lack of comprehensive simulation tools that provide a unified platform for vehicular networking and computing. Given the high cost and complexity of real-world vehicular experiments, such a tool is essential for designing and testing algorithms in this domain.

In light of these challenges, this dissertation makes the following contributions:

1. **Reputation-based Mechanism for VANETs Communications**: The first contribution is the design of a reputation-based mechanism that identifies malicious messages within VANETs. The mechanism works with standardized message formats and securely stores computed reputation scores using a Direct Acyclic Graph (DAG)-based ledger, ensuring tamper-resistant and distributed trust management.

2. **Creation of ETSI-compliant ITS-Message Dataset**: The second contribution involves generating the first ETSI-compliant dataset that includes both safety and non-safety messages exchanged in V2V and Vehicle-to-infrastructure (V2I) communications.

3. **Leveraging Time-Sensitive Networking (TSN) in Next-Generation Vehicles**: The third contribution relies on the emerging role of TSN as a technology for in-vehicle communications in next-generation vehicles [29]. TSN offers precise timing, synchronization, and deterministic communication capabilities. The idea is to leverage TSN to orchestrate coordinated message transmission schedules for out-vehicle communications, reducing the risk of message collisions and significantly improving the overall reliability of VANET.

4. **An ETSI MEC-compliant VC Architecture**: The fourth contribution is the proposal of VC architecture based on the ETSI MEC standard. This approach extends the computational capacity of MEC nodes by integrating resources provided by vehicles and other far-edge devices, exposed and made available in a standardized way. A custom simulation tool has been developed to model the volatility of devices in the resource pool, providing

a testing platform for engineers and researchers to experiment with applications in a VC context.

5. **Integration of MEC and O-RAN Standards in 5G Networks**: The fifth contribution is the extensive study conducted on the integration of the ETSI MEC standard in a 5G network compliant with the O-RAN specification. This study lays the foundation for a real testbed that demonstrates the feasibility of the VC paradigm exploiting the MEC components, as proposed in this dissertation. Additionally, this integration enables the deployment of user-related applications (i.e., MEC app) alongside those related to the RAN management (i.e., xApps and rApps). The study also enabled the development of a framework for creating near-real-time O-RAN-compliant applications, which play a central role in this integrated environment.

1.2 Related Publications

Parts of the work described in this dissertation come from following peer-reviewed publications:

- [30] Feraudo, A., Calvio, A., Bujari, A., and Bellavista, P., "**A Novel Design for Advanced 5G Deployment Environments with Virtualized Resources at Vehicular and MEC Nodes**." 2023 IEEE Vehicular Networking Conference (VNC). IEEE, 2023.

- [31] Feraudo, A. "**PhD Forum Abstract: Vehicular-based Support to Cooperative Edge Computing based Applications in Next-gen Networks.**" Proceedings of the 22nd International Conference on Information Processing in Sensor Networks. 2023.

- [32] Feraudo, A., Calvio, A. and Bellavista, P. "**A Novel OMNeT++- Based Simulation Tool for Vehicular Cloud Computing in ETSI MEC-Compliant 5G Environments.**" In Proceedings of the 13th International Conference on Simulation and Modeling Methodologies, Technologies and Applications. 2023.

- [33] Feraudo, A., Calvio, A., and Bellavista, P. "**Simulating and Validating Vehicular Cloud Computing Applications in MEC-enabled 5G Environments.**" Journal of Simulation Engineering. 2024.

- [34]Feraudo, A., Romandini, N., Mazzocca, C., Montanari, R., and Bellavista, P. "**DIVA: A DID-based reputation system for secure transmission in VANETs using IOTA.**" Computer Networks. 2024.

- [35] Feraudo, A., Maxenti S., Lacava A., Bellavista P., Polese M., and Tommaso M. " **xDevSM: Streamlining xApp Development With a Flexible Framework for O-RAN E2 Service Models.**" Proceedings of the 18th ACM Workshop on Wireless Network Testbeds, Experimental evaluation & Characterization. 2024.

I have also contributed to the following non-peer-reviewed publication that contributes to the work of this dissertation:

- [36] Feraudo, A., Andrea G., and Paolo B. "**Controlling Communications Quality in V2V Platooning: a TSN-like Slot-Based Scheduler Approach.**" Preprint. 2024.

During my PhD, I have also coauthored the following publication, that do not directly and primarily contribute to the work described in this dissertation:

- [37] Feraudo, A., Popescu, D. A., Yadav, P., Mortier, R., and Bellavista, P. "**Mitigating IoT Botnet DDoS Attacks through MUD and eBPF based Traffic Filtering.**" In Proceedings of the 25th International Conference on Distributed Computing and Networking. 2024.

2 Fundamentals

This chapter provides the concepts necessary to understand the architectures, technologies, and frameworks central to this dissertation. It begins with an overview of networking elements in vehicular networking environments (section 2.1), emphasizing European standards, and in 5G technology (section 2.2). The chapter then discusses the primary functional elements of the ETSI MEC standard (section 2.3) and investigates its integration within 5G-based scenarios. Next, it describes available vehicular computing paradigms and possible layering models (section 2.4). Finally, the chapter examines the main components of the O-RAN specification (section 2.5). Together, these concepts form the technological foundation upon which the research and contributions of this dissertation are built.

2.1 Vehicular Networks

VANETs are self-organizing networks where vehicles serve as mobile nodes or routers, facilitating communication over extended distances. Each vehicle is equipped with an OBU, allowing it to connect with nearby vehicles and establish network links dynamically. In such scenarios, RSUs are strategically placed along roadways to enhance service access and enable broader vehicle communication. As illustrated in Figure 2.1, VANETs support several communication modes, including V2V for direct data exchange between cars, V2I for interactions with roadside infrastructure, and Vehicle-to-everything (V2X), which extends communication to pedestrians, devices, and traffic systems, enhancing overall traffic management and safety.

Currently, two primary technologies promote VANET communications: IEEE 802.11p [5] and Cellular-Vehicle-to-everything (C-V2X) [38, 6]. The 802.11p standard operates at the physical and Medium Access Control (MAC) layers, while it utilizes the 802.11 wireless technology to enable wireless V2V and V2I communications. However, despite the good performance of this standard, it does not assure satisfactory reliability of message delivery. In 2019, the Internet Task Force (ITF) developed a novel V2X communication standard, known as IEEE 802.11bd. This standard is based on the IEEE 802.11ac (i.e., Wi-Fi 5), which makes it more powerful than its predecessor IEEE 802.11p. Specifically, it should guarantee twice the performance of IEEE

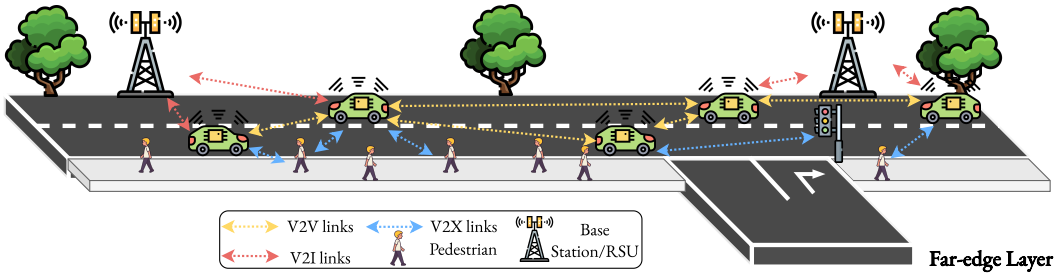


Figure 2.1: Vehicular ad hoc Networks Communication Modes

802.11 [39] in terms of throughput, latency, reliability, and communication range while guaranteeing backward compatibility with the previous standard. Furthermore, the use of Low-density Parity-check Coding (LDPC) and Midambles enables IEEE 802.11bd to achieve better reliability of message delivery compared to IEEE 802.11p, i.e., about 88% vs 75% [40].

On the other hand, C-V2X, developed by the 3GPP, leverages the existing cellular network infrastructure to offer a unified solution for V2V, V2I, and V2X communications. Initially introduced in Release 14 (Long-Term Evolution V2X) [38], C-V2X supported only broadcast communications. However, with the introduction of New Radio (NR) V2X in Release-16 [6], support was extended to unicast and groupcast communications. In such a scenario, vehicles exploit the PC5 interface for direct V2V communications independent of cellular networks, while the Uu interface is used to facilitate traditional cellular communications, enabling vehicles to receive information about road and traffic conditions.

The message specification for these technologies may vary across regions. For example, in the United States V2V communications are governed by the Wireless Access in Vehicular Environments (WAVE) standard [41], while in Europe, the ETSI group has established the ITS-G5 standard [42]. Both variants include messages with comparable information but different encodings, such as the Basic Safety Message (BSM) in WAVE and the CAM in ITS-G5.

ITS-G5: European Protocol Stack for V2X Communication

The set of standards provided by ETSI [42, 43, 44, 8], defines any vehicle or, more broadly, any device participating in the ITS system as an ITS station. This includes both road users, such as vehicles, and RSUs. Each ITS station operates according to the protocols at the various layers, as depicted in Figure 2.2. The protocol stack defined by ETSI [44] aligns with the Open Systems Interconnection (OSI) model, comprising four horizontal protocol layers and two vertical protocol entities.

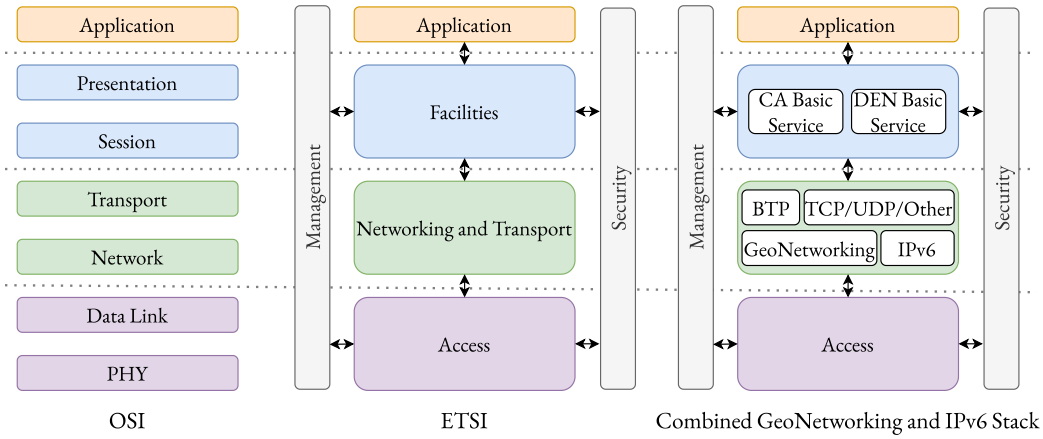


Figure 2.2: Intelligent Transportation System European Protocol Stack

The horizontal layers include the access layer, responsible for internal communications within the ITS station and external communications with other nodes. The networking and transport layer involves protocols that handle data delivery between ITS stations and other network nodes, ensuring reliable data exchange across the network. An instance of Basic Transfer Protocol (BTP) using the GeoNetworking protocol stack is provided in Figure 2.2. The GeoNetworking Protocol [43] leverages geographical positions for data packet distribution, relying on a geographical routing process to direct messages to specific geographic areas. Figure 2.2 also reports how BTP can be combined with IPv6. Next to this layer, the facilities layer offers essential functions such as data aggregation, storage, and message management, supporting the overall application functionality by handling message exchange and maintaining active connections.

The application layer focuses on the specific use cases and applications within an ITS scenario, such as traffic management or autonomous driving. Two vertical entities also support cross-layer functions: the ITS management entity, which configures the ITS station and ensures information exchange between layers, and the ITS security entity, which handles security services such as secure message transmission, identity management, and platform security.

European Standard Message Formats for V2X Communication

As mentioned previously, in Europe, the ETSI has released several standards [8, 19, 20, 45] for vehicular communication message formatting. Among these, the two primary formats are the Cooperative Awareness Message (CAM) [19] and Decentralized Environmental Notification Message(DENM) [20]. CAMs facilitate awareness among ITS stations by enabling both road users and infrastructure to share information about their position, dynamics, and characteristics. This information is transmitted periodically, with its management and processing handled by Cooper-

Table 2.1: CAM and DENM Message Content Examples

CAM Message Content		DENM Message Content	
Column Name	**Description**	**Column Name**	**Description**
source	stationID of the ITS-S generating the message.	**source**	stationID of the ITS-S generating the message.
destination	Destination of the message (null if broadcast).	**destination**	Destination of the message (null if broadcast).
messageID	Message type: (2) cam	**messageID**	Message type: (1) denm
refPositionLat	Latitude of reference point.	**actionID**	Event ID: stationID:seqNumber.
refPositionLong	Longitude of reference point.	**situation_infoQ**	Probability of event's existence.
refPositionAlt	Altitude of reference point.	**situation_eventType**	Describes the event type.
genDeltaTime	Time of CAM generation.	**location_heading**	Heading of the event.
isSpecialVehicle	True/False based on vehicle type.	**detection_time**	Time event was detected.
causeCodeIfSpecial	Describes emergency/safety event type.	**ref_time**	Time DENM was generated.
stationType	Type of the originating ITS-S.	**location_speed**	Speed of detected event.
		eventPos_lat	Latitude of the event.
		eventPos_long	Longitude of the event.
		eventPos_alt	Altitude of the event.
		termination	Termination: - Cancellation: By originating ITS-S - Negation: By another ITS-S
		stationType	Type of the originating ITS-S.

ative Awareness (CA) basic service. The CA basic service operates at the facilities layer, interacting with the ITS application layer to gather pertinent data for CAM generation and forward received CAM content for further processing. The specific content of these messages may vary based on the type of ITS station, such as whether it is a vehicle or a RSU. A use case demonstrating the application of CAMs is Cooperative Adaptive Cruise Control [8], an ITS application where continuous communication between a target vehicle and a subject vehicle enables the subject vehicle to adjust and maintain a reduced time gap with the target vehicle.

On the other hand, DENMs are used to transmit exceptional events detected by ITS stations, such as road hazards or abnormal traffic conditions. Upon detection of an event, an ITS station transmits a DENM to inform nearby ITS stations about the situation. Unlike CAMs, DENMs are not sent periodically; instead, they are generated and disseminated as long as the event is ongoing. The Decentralized Environmental Notification (DEN) basic service (see Figure 2.2 right stack) is responsible for constructing and delivering the DENM payload to the ITS networking and transport layer. These messages can also be initiated by the source of an event, for instance, in case of electronic brake light malfunctioning. One practical example of DENM application is in advanced Pre-Crash sensing systems [8], which provide information on detecting critical situations. Upon receiving a DENM, each vehicle can activate its Pre-Crash measures if it assesses that it is at risk.

Table 2.1 presents an overview of the message contents discussed in this section. It should be noted that field names have been adapted to align with the algorithms and solutions described throughout this dissertation.

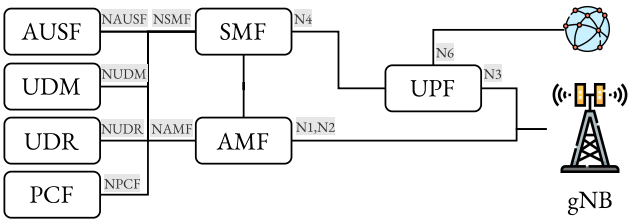


Figure 2.3: 5G Core Network Functions

2.2 5G New Radio RAN and 5G Core

5G, the fifth generation of mobile cellular networks, was formalized in December 2017 by the 3GPP [46], marking it as the successor to 4G networks. This section introduces essential aspects of the 5G NR RAN and 5G Core, focusing on the elements relevant to this dissertation.

5G New Generation-RAN architecture

The primary components of the Next-Generation RAN (hereafter RAN) are the User Equipment (UE) and Next Generation NodeB (gNB). The UE can be various mobile devices, such as smartphones, vehicles, or Internet of Things (IoT) devices like smart city sensors or cameras. The UE is responsible for bidirectional data transfer through the 5G network, ensuring correct bearer allocation and providing security measures like data encryption. On the other hand, gNBs supply radio coverage and are equipped with enhanced capabilities to meet 5G requirements, including ultra-low latency, increased bandwidth, and massive IoT support. Additionally, the gNB handles multiple devices and cells, manages radio resources, and schedules uplink and downlink data transfers over the radio interface.

The radio interface connecting the UE and gNB is the 5G Uu interface. Additional connectivity includes the textitXn reference points that allow inter-gNB communication, while connectivity to the core network is achieved through *N* reference points. For instance, the *N2* interface supports control plane data flow, while *N3* handles user plane traffic and data transmission. Figure 2.3 provides an overview of these reference points.

5G Cloud RAN

Service providers may deploy 5G base stations using a cloud-native approach named cloud RAN. This approach virtualizes radio access network functions to enable more flexible, scalable, and efficient deployments. This method shifts base station computing and storage capabilities to a centralized Central Unit (CU), responsible for non-time-critical processing activities of gNBs. The Distributed Unit (DU), located at the gNB side, exchanges traffic with the CU, while an

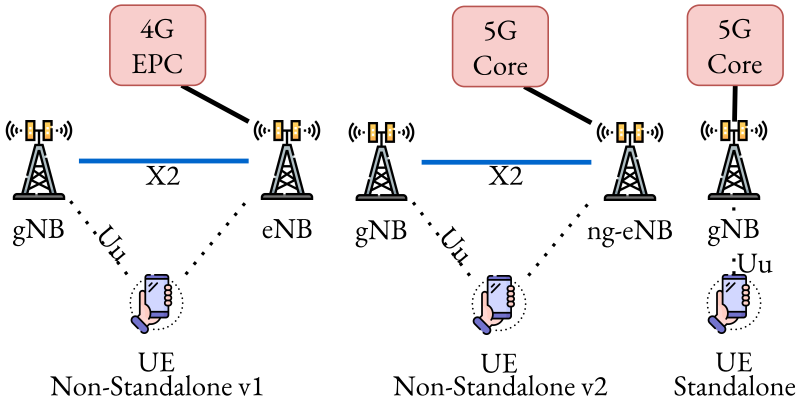


Figure 2.4: 5G Deployments

additional unit called Radio Unit (RU), responsible for radio transmission, is deployed at cell sites and connects with the DU.

5G Core

As already demonstrated, 5G architecture follows cloud-native development principles, with the 5G core comprising multiple independent network functions that handle functionalities such as mobility management, session management, traffic policy, and QoS control. While Figure 2.3 illustrates most of the network functions currently defined by 3GPP, this dissertation provides details of only three of them: the Access and Mobility Management Function (AMF), Session Management Function (SMF), and User Plan Function (UPF).

The AMF is responsible for managing the control plane interface from the RAN, including ciphering and integrity protection of UE registration management messages. Acting as a gatekeeper, it rejects registrations from UEs with invalid subscriptions and manages their mobility across base stations and AMF regions.

The SMF manages session creation, modification, and termination. It maintains the tunnel between the UPF and the gNB, handles UE IP address allocation, and responds to Address Resolution Protocol (ARP) and IPv4/6 neighbor solicitation requests based on its local cache. The SMF selects the UPF that will service the subscriber based on different parameters and provides traffic steering information to the UPF to route the traffic to the proper destination.

The UPF is the primary data plane network function within the 5G core. It performs packet routing and forwarding, may inspect packets, and optionally assigns IP prefixes as directed by the SMF. The UPF enforces redirection and traffic steering policies based on the SMF's instructions, ensuring QoS for both uplink and downlink transmissions.

5G NR Deployments

Mobile network operators will go through a transition phase before transiting completely to the 5G network architecture. Two strategies exist: Non-standalone (NSA) and standalone (SA) deployment. In NSA, the 5G gNB supports dual connectivity with a 4G Evolved Node B (eNB), allowing UEs to connect to both 4G and 5G networks. This setup provides 5G services to users while utilizing the 4G Evolved Packet Core (EPC) for control connectivity, as shown in Figure 2.4. Another NSA approach involves the Next-Generation eNB (ng-eNB) and the 5G Core, where the gNB communicates with the core via dual connectivity with the ng-eNB.

All experiments in this dissertation are conducted on a 5G SA network, where gNBs connect directly to the 5G core without dual connectivity across base stations (deployment on the right in Figure 2.4).

2.3 Multi-access Edge Computing

The MEC standard, developed by the ETSI, responds to the need for a standard for a virtualized, multi-tenant environment at the network's edge. It offers a standard way to distribute virtualized resources throughout the network by supporting the deployment of contextualized MEC Applications (MEC Apps) near data sources and end-users while also enabling seamless integration with cloud infrastructure. These applications, which can be provided by vendors, service providers, or third parties, are deployed across multi-vendor platforms at the edge, creating a distributed cloud continuum.

Figure 2.5 illustrates the key functional components included in the reference architecture [11]. The architecture is organized into two primary layers: the system and the host layer. Each element within these layers is linked to the others via reference points, allowing the exchange of standardized information, i.e., management (Mm), external (Mx), and MEC Platform-related (Mp).

The system layer enables interactions between users and the overall infrastructure, handling the underlying elements and managing MEC-Hosts (MEC-Hs) distributed across the network. Device Apps (D-App) are components deployed as applications in traditional UE devices like vehicles, which can interact with the MEC infrastructure via the User Application LifeCycle Management Proxy (UALCMP). The latter helps manage all the procedures to instantiate, delete, and relocate MEC Apps upon client requests. The core component at this layer is the MEC Orchestrator (MEC-O), which has a holistic view of all MEC-Hs across the network, including available resources, services, and topology. The MEC-O selects the most suitable host where to deploy the MEC-App requested. The choice of the best host can be based on different algorithms favoring the specific requirements of each requested service. Finally, this layer includes the Customer Fac-

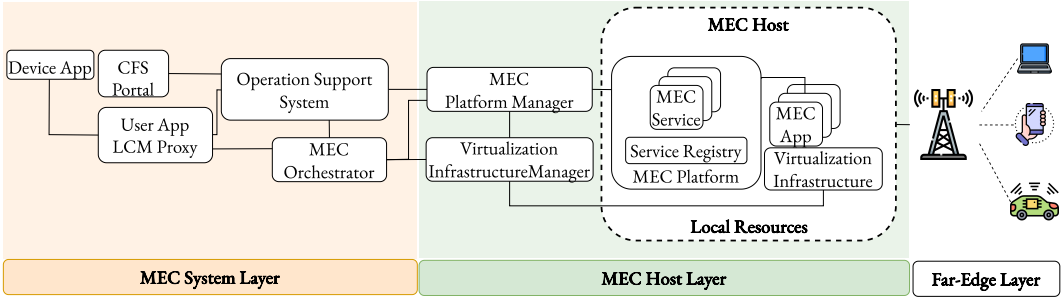


Figure 2.5: ETSI MEC Reference Architecture

ing Service (CFS) portal, allowing third parties to select and order a set of MEC Apps that meet their needs.

The host layer comprises the key functional elements responsible for managing the virtualized environment and its resources, including storage, computing, and networking. These components facilitate the instantiation, deletion, and control of MEC Apps running on the infrastructure. At the core of this layer is the Virtualization Infrastructure Manager (VIM), which handles the virtualized resources within the MEC-H's Virtualization Infrastructure (VI). All operations required to prepare the infrastructure for a new MEC App deployment are handled by the VIM. The MEC-H, which serves as edge node hosting MEC Apps, enable these applications to interact with standard services like Location Service, Radio Network Information (RNI), and Application Mobility Service (AMS), through the MEC Platform (MEC-P). The platform maintains a service registry with information about service endpoints, facilitating integration between several MEC-Hs. The MEC Platform Manager (MEC-PM) serves as an intermediary, ensuring that the MEC-O is updated on MEC Apps' status by forwarding relevant application events. This component also acts as an administrator, managing configurations, handling fault reports, and monitoring performance metrics from the VIM.

Deploy ETSI MEC in 5G Networks

Within the realm of 3GPP 5G networks [46, 6], MEC deployments are a key focus for optimal performance [47]. The MEC standard is designed with the intent of providing context-aware services in 4G/5G deployments, delivering information related to the RAN and UE (e.g., RNI service). The reference architecture is also compatible with the cloud-native approach on which the 5G network relies, where network functions consume or produce services.

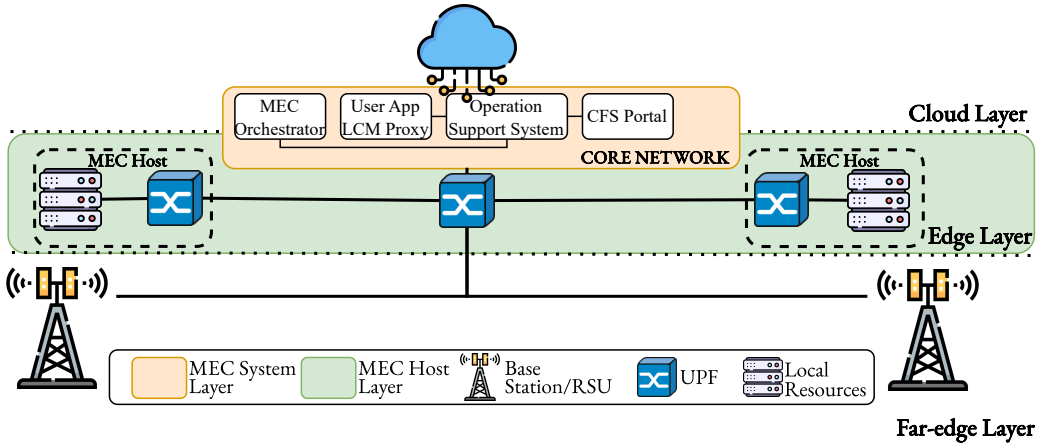


Figure 2.6: MEC Host in 5G Network

A key component in this integration is the UPF, the main data plane network function in the 5G Core (see section 2.2). User data traffic flows from the UE to the gNB, passing through the user plane before exiting the data network. From a MEC perspective, the UPF functions as a distributed and configurable data plane. As a result, in certain deployments, the UPF can be directly integrated into the MEC architecture, facilitating localized data handling at the network edge.

Another 5G Core network function relevant to this integration is the SMF (section 2.2). Given that MEC services may need to be accessed by other edge nodes, the SMF is crucial in ensuring traffic steering toward the correct MEC Apps by configuring the UPF rules accordingly.

As mentioned in the previous section, MEC-Hs is logically deployed at the edge of the network, meaning the UPF is responsible for steering the user plane traffic towards the targeted MEC Apps in the data network. Network operators have multiple options to decide where the MEC-Hs should be deployed in the network[47]. This dissertation focuses on deployments where the MEC-H and the local UPF are co-located with a network aggregation point, as depicted in Figure 2.6.

2.4 Vehicular Computing Paradigm

VCC relies on the principle that future vehicles have enough computation power to serve as computation platforms for edge-based services. The early investigations into this paradigm were led by [15] and [14], both recognizing the potential of modern vehicles equipped with powerful onboard computers, heavy storage, and a range of sensors. Olariu *et al.* [15] defined VCC as a collaborative way to share resources among vehicles to solve problems that would otherwise require a significant amount of time with a more traditional centralized architecture, in particular for

context-specific applications. Gerla in [14] defined this paradigm as a platoon of vehicles that keep the information gathered by sensors locally and share it solely with other vehicles. This approach helps address the significant technical challenges posed by the vast amounts of data generated within vehicles for network infrastructure. With the proliferation of sensors in vehicles [1], it is predicted exponential growth in the data traffic generated by vehicles [12]. Therefore, Vehicular Cloud Computing proves to be crucial in mitigating network congestion by facilitating data pre-processing directly among groups of vehicles [14].

Complementing VCC, VEC paradigm integrates vehicular networks with edge computing infrastructure, bringing computational resources closer to vehicles. This proximity reduces latency and improves application availability through V2I communications [3, 7]. In such an environment, vehicles can offload computation- or latency-sensitive tasks to edge servers, often hosted by RSUs, without needing to access the core network, enhancing efficiency and responsiveness.

Both paradigms generally rely on the often-underutilized computational resources included in vehicles. Resources included in parked or congested vehicles often remain idle for extended periods, leaving substantial computational power unused. Conversely, there are situations where a vehicle lacks the necessary resources to execute a task, while nearby vehicles may have surplus resources that could fulfill the demand. To manage this, vehicular edge architectures typically organize resources into layers. As detailed in [3], there exist multiple layering models. The two-layer architectures generally include an edge layer supporting vehicular applications and a vehicular layer aggregating vehicle computing resources. Three-layer architectures add a cloud/controller layer, which provides significant computational resources for processing complex data and orchestrating edge resources. Some architectures even utilize a four-layer model that abstracts resource orchestration into an independent layer, thus introducing more complexity.

Building on these concepts, this dissertation integrates both VCC and VEC paradigms, encompassing all available resources across the cloud continuum—cloud, edge, and far edge—into a unified framework called VC. Figure 1.1 shows the resources layering at the core of this dissertation.

Despite its potential benefits, the VC paradigm also introduces several challenges that must be addressed. One key issue is distributed ownership, where each vehicle is owned individually, leaving resource-sharing decisions to the vehicle's owner. Another challenge is high node mobility, which makes it difficult to predict the vehicular residency times in the cloud even when clouds are formed using resources of stationary cars within a parking lot [17]. Device heterogeneity, as vehicles are manufactured by different companies [2]. Finally, due to the sensitive information shared among cars, security, authentication, and privacy represent critical open issues in such a context [17, 3, 2].

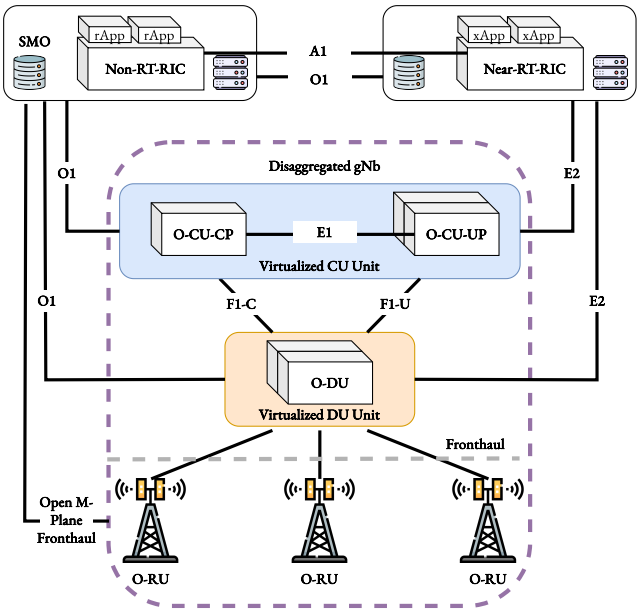


Figure 2.7: O-RAN Architecture Overview

2.5 Open Radio Access Network

The O-RAN Alliance [48] introduces a new RAN paradigm based on the concept of openness, marking a significant shift in how networks are deployed and optimized. This approach enables gNBs to expose data and analytics, facilitating closed-loop control and automation. As illustrated in Figure 2.7, the gNB is disaggregated into different functional units: the CU, DU, and RU. The CU is further split into two logical components—one for the control plane (CP) and the other for the user plane (UP)—which can be deployed at different locations within the network, offering flexibility in resource management. In this direction, the O-RAN Alliance detailed Fronthaul specifications, defining open interfaces that support the 7.2x split configuration, chosen from those specified by 3GPP. This configuration divides the physical layer between the RU and DU, with the RU handling phy-low functions and the DU handling phy-high functions. Furthermore, O-RAN extends the 3GPP-standardized gNBs by supporting RAN Intelligent Controllers (RICs) through open interfaces that stream telemetry from the RAN and apply dynamic configurations in real-time.

The current O-RAN specifications include two instances of the RIC, operating at different time scales and on different parameters. The Non-Real-Time (RT) RIC, hosted in the network Service Management and Orchestration (SMO), supports control loops with a time scale higher than 1 s, and enforces policies or high-level configurations on the system. The Near-RT RIC per-

forms radio resource management by directly tuning the configuration parameters of the RAN at a time scale between 10 ms and 1 s. These RICs host custom applications known as rApps and xApps, respectively, which implement the closed-loop control logic.

The O-RAN alliance has also defined technical specifications describing the open interfaces connecting the O-RAN components. These interfaces make the RAN *open*, as they facilitate the exposure of data analytics and telemetry and enable several control and automation procedures through the RICs. As illustrated in Figure 2.7, the E2 interface connects the Near-RT RIC to the RAN nodes, enabling near real-time closed-loop control. Meanwhile, the A1 interface links the Non-RT RIC and Near-RT RIC, enabling control, policy enforcement, orchestration, and deployment of intelligent models at non-real-time scales. The O1 interface allows the Non-RT RIC to manage and orchestrate other RAN components' network functions.

To ensure vendor interoperability across RAN and Near-RT RIC implementations, the O-RAN Alliance Working Group (WG) 3 has developed specifications for multiple protocols that define E2 interface [49]. The first specification is the E2 Application Protocol (E2AP) [50], which governs general management operations, such as establishing or disconnecting a gNB from the Near-RT RIC, and provides a list of supported RAN functions. The E2 Service Model (E2SM) specifications [51] define the semantics of interactions between xApps and gNBs. Each service model is featured in a document extending the general specification of the E2SM with parameters related to the use case of interest. Examples of these extensions are the E2SM Key Performance Measurement (KPM) [52], for data collection, and the E2SM RAN Control (RC) [53], which defines the protocol to perform Radio Resource Management (RRM) actions such as traffic steering through handover management.

This dissertation focuses on the Near-RT RIC and its associated xApps, as they play a critical role in radio resource management and provide real-time access to radio-related metrics. The following chapters will explore how this component of the O-RAN architecture can be leveraged to integrate ETSI MEC in a 5G network and build a testbed that supports advanced application scenarios for next-generation networks.

Near-RT RIC implementations

The RIC platform itself has been subject to several development efforts, primarily for scaled-down versions of the Near-RT RIC [54], such as the O-RAN Software Community (OSC) reference implementation [55] and the Open Air Interface (OAI) FlexRIC [56]. The OSC reference implementation provides a micro-service-based architecture for the Near-RT RIC, including various components, such as internal message routing via RIC Message Router (RMR), E2 termination, a subscription manager, a network information base database, and a shared data layer API, among

others. Each of these components operates within a Kubernetes cluster, ensuring scalability and efficient resource management. Besides the Near-RT RIC implementation, the OSC project also offers an xApp framework [57] that uses the RIC RMR to manage the message exchange with the remaining components of the RIC platform.

Another significant development is OAI FlexRIC, a Software Development Kit (SDK) for software-defined RAN controllers that supports an E2-compatible protocol. FlexRIC includes an iApp component that implements E2SMs, exposing data to xApps, along with a server library that manages connections between xApps and dispatches E2AP messages to E2 Nodes. The FlexRIC approach relies on a monolithic architecture, where all Near-RT RIC operations are included in a single component, decreasing the flexibility of the RIC platform.

This dissertation uses the O-RAN components from the OSC project, as they provide a reference implementation closely aligned with the O-RAN Alliance's architecture and specifications. This ensures greater compatibility and consistency with ongoing industry developments in the O-RAN ecosystem.

3 Supporting V2X Communications Reliability

As described in section 2.1, the content of vehicular messages can differ due to varying specifications in different regions [41, 42]. This Chapter, more broadly, this dissertation, focuses exclusively on the European message formats as defined by ETSI, focusing on ensuring the integrity of the transmitted data. Maintaining the trustworthiness of this information is critical, as the content of these messages is vital for upholding traffic safety in vehicular scenarios. Additionally, it is essential to mitigate interference and data loss, which could compromise the safety of these environments.

In response to these challenges, this Chapter investigates two solutions. First, it provides the design and implementation of a reputation-based system to reliably identify malicious messages within VANET environments while safeguarding vehicle owners' privacy. Second, it introduces a solution that extends modern in-vehicle communication systems, leveraging TSN [29], to external vehicular communications (V2X). This extension aims to bring the benefits of TSN's deterministic networking, low latency, and high reliability to V2X scenarios, enhancing data delivery and reducing packet collisions in vehicular networks. The Chapter also provides the results obtained during the experimentation of these solutions to validate their effectiveness in simulated yet realistic settings.

3.1 Securing V2X Transmission with Reputation Systems

VANET scenarios present a multitude of new security challenges [22, 23, 24, 3]. According to [23], two categories of attack surfaces can be identified in VANET:

1. **VANET attacks**. These comprise messages exchanged between vehicles and between vehicles and infrastructure, including environmental information (speed, direction, abnormal road events, etc.). For example, an attacker may exploit vulnerabilities to forge messages containing false or misleading information, potentially leading to severe consequences such as car accidents or misrouted vehicles. Additionally, passive attacks, such as packet eaves-

dropping, can be carried out to gather sensitive information about vehicles and traffic in specific areas.

2. **In-Vehicle attacks**.These attacks target the communication between in-vehicle hardware components, potentially compromising critical systems. For instance, they can remotely unlock vehicle doors, disable the braking system, or impair sensors essential for autonomous driving, posing significant safety risks.

This dissertation focuses on addressing VANET attacks, proposing a solution called Decentralized Identifiers (DID)-based reputation system for secure transmission in VAnets (DIVA), designed to offer a reliable solution to identify malicious messages. DIVA requires every vehicle to have a unique identifier and share road information with other entities within the network. Vehicles are identified with Decentralized Identifiers (DIDs), while their participation is regulated using Verifiable Credentials (VCreds). A DID consists of three parts: a Uniform Resource Identifier (URI), a specific DID method identifier, and a method-specific DID identifier. DIDs eliminate the need for identity providers and centralized authorities, allowing entities to prove ownership by using private keys corresponding to the embedded public keys in the DID document. Verification is achieved by accessing the public DID Document, typically shared through a verifiable data registry often implemented via Distributed Ledger Technologys (DLTs). DLTs are decentralized systems that replicate data across multiple nodes, ensuring tamper-proof records without a central authority. They rely on Peer-to-Peer networks and cryptographic consensus to prevent single points of failure. The two main types of DLTs are blockchain, which links blocks of data using hash pointers to detect tampering, and DAG, a directed graph structure without cycles that provides a more flexible alternative to traditional blockchains.

On the other hand, VCred [58] is a standardized data structure for representing cryptographically verifiable and tamper-proof claims. In the VCred ecosystem, holders manage VCreds, issuers (e.g., the Ministry of Transport) create them, and verifiers use them to establish trust, such as when an RSU collects vehicle data.

In the proposed system, each DID starts with a default reputation score, which is then adjusted based on the history of exchanged message contents and their truthfulness. Reputation scores are stored in the IOTA Tangle [59], a DAG-based ledger spread across multiple edge nodes covering a specific geographical area, e.g., a city. The computation of reputation scores occurs at edge nodes avoiding computation overhead and delays on the participating vehicles.

To evaluate the effectiveness of the proposed reputation system, this dissertation provides a dataset including safety and non-safety V2X messages compliant with the ETSI standard. The

dataset has been further augmented by introducing controlled instances of malicious messages, allowing for testing how DIVA responds to misbehaving vehicles.

The contributions of this section are as follows:

- It provides design and implementation details of a novel reputation scheme for VANET, with full compliance with ETSI standards;
- It describes an extensive ETSI-compliant communication dataset generated to measure the performance of the proposed reputation scheme.

Additionally, this section surveys the state-of-the-art reputation systems for VANET, focusing on blockchain-based approaches. It also reviews existing datasets on VANET communications to provide context and relevance for the proposed solution.

3.1.1 Related Work

This section aims to provide a literature review of reputation schemes based on blockchain and DAG - the two main types of DLT employed in vehicular scenarios. In addition to exploring these reputation schemes, this section also provides a comprehensive overview of existing datasets related to vehicular communications, highlighting their use in such a context.

DLT-based Reputation Systems

This state-of-the-art analysis distinguishes between blockchain-based and DAG-based reputation schemes in vehicular communication networks.

Reputation schemes based on blockchain have been widely explored in the context of vehicular communication networks. Li *et al.* [60] introduced a mechanism named BDRA to secure registration and authentication using a double-layer blockchain and DIDs. The first layer includes authorized RSUs, while the second one categorizes vehicles considering the corresponding RSU coverage area. The RSU and the vehicles collaborate to form a consortium blockchain in each area. DIDs are leveraged to reduce reliance on third-party intermediaries for the registration phase and implement the reputation feedback mechanism. This mechanism updates reputations relying on messages sent by vehicles. However, there is no indication of how the truthfulness of these messages is verified.

Similarly, Fernandes *et al.* [61] introduced BRS4VANET, a decentralized reputation system leveraging a consortium blockchain and smart contracts. In this system, RSUs calculate and store vehicles' reputations blockchain based on feedback from other vehicles. To ensure privacy, vehicles utilize pseudo-anonymous certificates, which are revoked if their reputation falls below a

predefined threshold. However, the authors do not outline the certificate revocation process in this work.

Lu *et al.* [62] proposed BARS, a blockchain-based anonymous reputation system, to increase the security of VANETs by preventing the propagation of malicious messages. BARS safeguards user privacy through pseudonyms generated from public keys. The proposed reputation system evaluates both direct and indirect interactions among vehicles, leveraging blockchain technology to securely and immutably store vehicle reputation certificates alongside their public keys. In this work, the authors do not address how the system verifies the truthfulness of messages.

Yang *et al.* [63] proposed a blockchain-based trust management system for VANETs. Each vehicle generates a reputation score for each received message based on a Bayesian Inference Model. In this approach, each vehicle evaluates received messages using a Bayesian Inference Model to generate a reputation score. These scores are then sent to the RSU, which aggregates reputation values from nearby vehicles and constructs blocks for the blockchain. RSUs are responsible for maintaining the blockchain and competing to append new blocks. The authors use vehicle identity numbers for identification, potentially compromising privacy.

Reputation schemes based on DAG represent an emerging research direction, offering advantages such as scalability, fast transactions, and low fees. Li *et al.* [64] proposed a partitioned DAG-based ledger for managing vehicular reputation in VANETs, focusing on local interactions where information remains relevant only to nearby vehicles for a limited duration. To establish trust, a node extracts interaction information from transactions and uses it to calculate the reputation of other nodes according to the situation. However, as the number of transactions increased, the approach became impractical. To mitigate this, the authors introduced a data structure for intermediate reputation calculations, which, while addressing the issue, introduced additional overhead and impacted scalability. Their approach primarily focuses on estimating and updating vehicle reputations but overlooks vehicle identification and communication, which are critical concerns to enhance communication security in VANETs.

Li *et al.* [65] presented a DAG-based reputation mechanism designed to realize the authenticity, immutability, and accessibility of all vehicle reputations while preserving their privacy. In the proposed system, the reputations of vehicles are used to determine the degree of protection of their privacy when a peer captures and uploads an image of a traffic accident on the DAG for mutual supervision. Although reputation is intended to be the core aspect of their proposal, the authors primarily focus on privacy concerns, providing limited details on how reputation scores are calculated and updated.

To extend the tamper-proof capability of blockchain to large-scale vehicle networks, Du *et al.* [66] proposed an information-sharing approach based on a DAG. To mitigate chain attacks,

they designed a reputation-based rate control strategy, where reputation is used to identify reliable vehicles that participate in voting to reach consensus and regulate the number of published transactions.

A Review of Data Collection in Vehicular Networks and Sensor Systems

Publicly available datasets are crucial for validating and benchmarking innovative solutions across real-world scenarios. This analysis focuses on datasets capturing data from vehicular systems, including in-vehicle and external-vehicle networks. The former primarily includes datasets providing data from the Controller Area Network (CAN) bus, a robust in-vehicle communication protocol that enables real-time communication between vehicles Electronic Control Units (ECUs). On the other hand, external-vehicle datasets regard wireless communication parameters, particularly in the context of V2X communications.

In-vehicle data collection offers insight into how vehicles communicate internally, making it a key area for security testing and anomaly detection. For instance, Song et al. [67] created a fully labeled in-vehicle dataset to demonstrate their intrusion detection algorithm for in-vehicle networks. They used two Raspberry Pi devices: one for logging CAN traffic via the OBD-II port and the other for injecting fabricated messages to simulate attacks. Four types of attacks were introduced: Denial of Service (DoS), Fuzzy attack, Gear Spoofing, and RPM Spoofing. Recently, Jeong et al.[68] generated seven CAN datasets by reading CAN bus data using a Kvaser Memorator Pro 2xHS device connected through the OBD-II port of a Hyundai LF Sonata 2017. Six of these datasets were collected during urban driving in Seoul, South Korea, and one was captured while the vehicle was stationary. In another study, Han *et al.* [69] collected CAN messages from three different vehicle models. The collected data were categorized into two types: normal driving data without attacks and abnormal driving data collected during an attack. In [70], the authors surveyed existing datasets and introduced the ROAD dataset, which features advanced attack types like stealthy fabrication attacks, providing more sophisticated testing scenarios for CAN intrusion detection system.

One of the early efforts to gather real-world V2X data was the European iTETRIS project [71]. The dataset included mobility data and IEEE 802.11p V2I communication traces. Cruz *et al.* [72] collected V2V communication data, GPS, inertial, and WiFi data over a 12 km road trip under typical traffic conditions to enhance vehicle localization accuracy.

Focusing on the U.S. standard, Malebary in [73] provided a dataset including BSM messages exchanged in V2V and V2I communications based on IEEE 802.11p, in regular and jammed condition. Similarly, the data provided in [74]captured complex-baseband samples of V2V communication in the presence of jamming signals, including the broadcasting of BSMs. Additionally,

Boban *et al.* [75] collected data on two V2X use-cases: teleoperated driving and local dynamic map update. Their dataset includes uplink and downlink throughput, location data, vehicle speed, and interference information to evaluate their solution enhancing QoS in V2X applications. Hernangómez *et al.* [76] expanded on these efforts by providing data collected in mixed urban and highway environments, including cellular and sidelink communications. Their dataset followed CAM regimes, covering packet size, transmission rate, and modulation and coding schemes, further enhancing the study of communication between vehicles in various real-world scenarios.

Discussion

Most existing reputation-based solutions for VANETs do not consider the accuracy of the information embedded within standard-defined message structures. Moreover, these systems often calculate reputation scores directly on vehicles, which requires processing message contents during exchanges, leading to notable delays.

Besides these limitations, reputation-based schemes face technical challenges due to the dynamic nature of VANET environments. The use of temporary identifiers (such as pseudonyms, as referenced in [77]) for vehicle identification, combined with the frequent changes in network topology within a VANET, pose challenges in the definition of reputation scores and the collection of adequate vehicle information. Furthermore, the computation and propagation of reputation scores require time, which can be a significant concern for VANET-based applications with latency-sensitive requirements [3].

Assessing reputation systems in a VANET scenario requires evaluating the information exchanged during communication. However, most existing datasets focus on radio-related data or non-standard content rather than message specifics [73, 75, 76]. Even previous works in this domain, such as [78], provide analysis related to message length and transmission frequency but lack actual message content.

To address these issues, the proposed solution, DIVA, leverages DID for managing vehicle identities and VCred to regulate participation. Unlike existing systems, in DIVA, the computation of reputation scores occurs at edge nodes without incurring computation overhead and delays on the participating vehicles. The system's performance has been rigorously tested using a novel dataset generated from a simulated scenario [79]. It includes safety and non-safety messages exchanged in V2V communications, representing the first attempt to create an ETSI-compliant dataset focused on message content. To test DIVA under realistic conditions, the dataset has been carefully manipulated by introducing controlled malicious messages, enabling the system to be evaluated on its ability to detect and respond to vehicle misbehavior effectively.

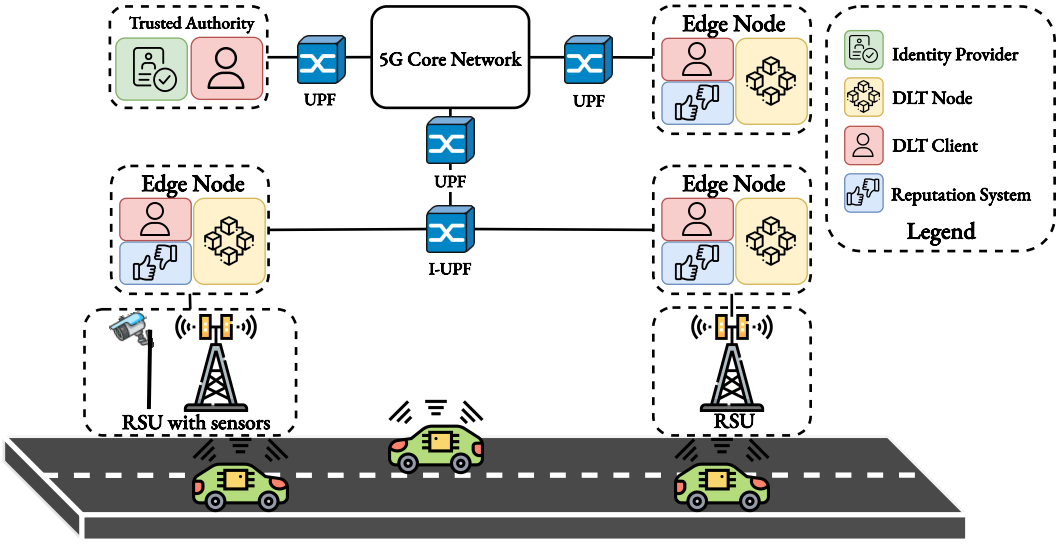


Figure 3.1: DIVA Deployment Scenario

3.1.2 DIVA Deployment Scenario

The scenario in which DIVA operates involves a network of edge nodes interconnected via a 5G core network, utilizing the IEEE 802.11p standard for both V2V and V2I communications. In this context, reputation-based information on IOTA is maintained across multiple edge nodes. An edge node serves as the entity providing computational resources in the form of a small data center located at the network edge, i.e., close to car drivers [80, 11, 30]. The placement of these nodes is determined by network operators according to their existing infrastructure, as outlined in section 2.3.

In a VANET scenario, edge nodes may act as collection points for messages from vehicles and perform computations to determine the trustworthiness of each participant. Edge nodes update the vehicle reputations through transactions stored in the ledger. Furthermore, it is assumed that they are designed to be resilient against tampering or unauthorized access.

Each edge node also maintains an *internal table* that holds the reputation associated with the vehicles, enabling them to respond quickly to reputation requests without querying the ledger. This organization minimizes the computation overhead on the vehicle side since participants in the VANET can directly leverage these data to assess the trustworthiness of other vehicles.

As previously noted, vehicles are uniquely identified using DIDs, assigned and registered on a ledger by a Trusted Authority (TA). As illustrated in Figure 3.1, the TA consists of an Identity Provider, such as the Ministry of Transport in Italy or an authorized inspection center, responsible for periodically verifying vehicles' compliance with legal standards by detecting any physical

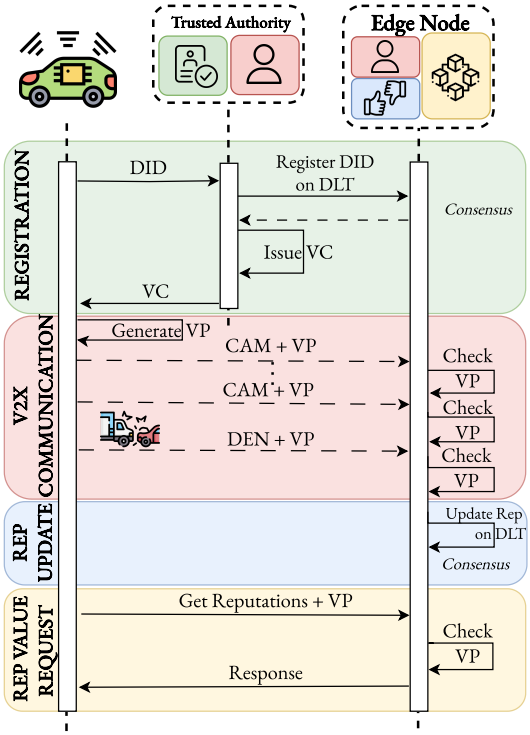


Figure 3.2: DIVA workflow

tampering or modifications. Upon completing these inspections, the Identity Provider issues a VCred for the vehicle's DID, certifying its eligibility.

Finally, RSUs are considered integral and trusted components of the infrastructure and, thus, play a central role in verifying message reliability. The presence of an RSU, providing road-related data, can assist in identifying malicious vehicles when their contributions inherently deviate from that of the RSU. However, it is worth noting that not all RSUs are equipped with sensors for monitoring road-related events; some may function as relay nodes, extending vehicles' communication range and facilitating communication with infrastructure components. Therefore, RSUs providing trusted road data are not uniformly distributed across all roadways. In situations where RSU sensors are absent, and the RSUs operate as relay nodes, the system assumes that most vehicles are benign.

3.1.3 DIVA: A DID-based Reputation System for secure VANETs

As illustrated in Figure 3.2, the DIVA operations defined in the current version of the system can be classified into registration, V2X communication, and reputation update.

Registration

Each vehicle must first register a DID on the ledger and have a VCred issued by a TA, which is responsible for verifying that the vehicle is compliant with the law and has not been stolen or altered. Upon completing the necessary checks, the vehicle generates a DID and submits it to the TA. This process guarantees that only the holder of the DID maintains complete control over its associated information. The TA interacts with the ledger to record the DID and the corresponding DID document. Next, the TA issues a VCred that verifies the successful registration of the DID and returns it to the vehicle. With the VCred in hand, the vehicle can create a Verifiable Presentation (VP), which will be included in each communication. The VP is proof of the vehicle's identity and ownership of the issued VCred.

V2X Communications

This system adheres to the ETSI message formats outlined in section 2.1 for V2X communications. Additionally, the system relies on the GeoNetworking Protocol [43] for message dissemination. For CA basic service message forwarding occurs through Single-Hop Broadcasting, where messages are sent only to one-hop neighborhoods. In contrast, the DEN basic service employs GeoBroadcasting for packet forwarding. In this approach, packets are transmitted hop-by-hop until they reach the designated destination area specified in the packet. Once the packet reaches the destination area, nodes within that region begin rebroadcasting it to ensure its effective dissemination.

Reputation Update

As section 3.1.2 mentions, the scenario assumes that each edge node receives the transmitted messages within the VANET. As shown in Figure 3.2, upon receiving a DENM, the edge node verifies the VP, evaluates the received information, and proceeds with the reputation update of the involved vehicles.

This process can be structured into three key steps: *reputation score computation*, *reputation score update*, and *reputation score storage*.

Reputation Score Computation. Information from RSUs equipped with traffic monitoring sensors serves as the ground truth in the DIVA system. These RSUs validate the authenticity of vehicle-generated messages and help assign reputation scores. As discussed in section 3.1.2, not all RSUs are equipped with sensors; some function solely as relay nodes to enhance communication, rendering them incapable of providing traffic condition data. To overcome this limitation, DIVA incorporates an outlier detection algorithm to identify vehicle-generated messages, safety-related,

Algorithm 1 Reputation Computation

Input: $\tau_{eventType}, \tau_{sitQuality}, \tau_{cam}, \omega_{rsu}, \omega_{msg}, defScore, m$
Output: $repScores$
Require: $m_{age} \geq \tau_{eventType}; m_{sitQuality} \geq \tau_{sitQuality}$

$rsuScore \leftarrow (\omega_{rsu} * defScore)$
$msgCohScore \leftarrow (\omega_{msg} * defScore)$

if $initialReputationExists(m_{source})$ **then**
  $repScore \leftarrow loadInitalRepScore(m_{source})$
else
  $repScore \leftarrow defaultRepScore$
end if
if not $messageInsideEdgeArea(m)$ **then**
  return
end if
if not $messageCoherentWithRSU(m)$ **then**
  $repScore \leftarrow repScore - rsuScore$
end if
$camCoh \leftarrow computeCAMCoherency(m, \tau_{cam})$
if $camCoh < 10$ **then**
  $repScore \leftarrow repScore - msgCohScore$
else
  if $10 \leq cam\_coh \leq 30$ **then**
    $repScore \leftarrow repScore$
  else
    $repScore \leftarrow repScore + msgCohScore$
  end if
end if
$similarEvents \leftarrow findSimilarEvents(m, \tau_{eventType}, \delta_{eventType})$

if $len(similarEvents) > 2$ **then**
  $repScore \leftarrow repScore - msgCohScore$
else
  $repScore \leftarrow repScore + msgCohScore$
end if
$updateRepScore(did_m, repScore)$ ▷ Applying eq.(3.1)

Algorithm 2 CAM coherency

Input: $denm, \tau_{cam}$
Output: $camCoherency$
function COMPUTECAMCOHERENCY($denm, \tau_{cam}$)
  $cams \leftarrow loadCam(denm_{source}, \tau_{cam})$
  $camCoh \leftarrow 0$
  for all cam **in** $cams$ **do**
    $distance \leftarrow getDistance(denm_{evPos}, cam_{refPos})$
    if $distance \leq \delta_{eventType}$ **then**
      $camCoh \leftarrow camCoh + 1$
    end if
  end for
  return
  return $(\frac{camCoh}{\text{len}(cams)}) \times 100$
end function

Algorithm 3 DENM coherency

Input: $denm, \tau_{eventType}, \delta_{eventType}$
Output: $similarEvents$
function FINDSIMILAREVENTS($denm, \tau_{eventType}, \delta_{eventType}$)
  $centroidsDenms \leftarrow loadCentroids()$
  for all el **in** $centroidsDenms$ **do**
    $eventType \leftarrow el[eventType]$
    $\tau_{centroid} \leftarrow el[time]$
    $\delta_{centroid} \leftarrow el[space]$
    if $eventType = denm_{eventType}$ **and** $abs(\tau_{centroid} - denm_{detTime}) \leq \tau_{eventType}$ **and** $distance(\delta_{centroid} - denm_{eventPos}) \leq \delta_{eventType}$ **then**
      $el[time] \leftarrow \frac{\tau_{centroid} + denm_{detTime}}{2}$
      $el[space] \leftarrow centroid(\delta_{centroid}, denm_{eventPos})$
      return $element[denms]$
    end if
  end for
end function

that deviate significantly from the average. Following this evaluation, a reputation score is assigned to the DID responsible for generating that message.

Each DID is linked to a reputation value $r_{DID} \in [0, 1]$, representing the vehicle's overall trustworthiness. The outlier detection algorithm, defined in Algorithm 1, requires the freshness of the messages $\tau_{eventType}$, which depends on the type of event leading to its dissemination and the situation information quality above a certain threshold, $\tau_{sitQuality}$. These data are extracted directly from the message content, specifically from the fields *situation_infoQ* and *situation_eventType*, as detailed in the DENM column of Table 2.1. For example, in the DENM structure, the *situation_infoQ* field ranges from 0 to 7, indicating the quality level of information vehicles provide. This may be influenced by the condition of the sensor that gathers that information. Moreover, the message's generation time is also encoded in the CAM and DENM structures, which helps identify outdated messages. Messages that are too old or of low quality are excluded from the reputation calculation, as they are typically ignored by other vehicles within the VANET.

After verifying message quality and freshness, the algorithm checks whether the event's location (*eventPos_lat/long/alt* Table 2.1), indicated in the message m, falls within the area managed by the edge node using the function $messageInsideEdgeArea(m)$. Next, it compares the con-

tent of the message with the information provided by the RSU and assesses its coherence with the CAMs generated by the same source. Only CAMs within a time window, defined by a threshold τ_{cam}, are considered. This time window can be adjusted according to the dynamic nature of the VANET. For instance, in a highway scenario where vehicles move faster, older CAMs might not reflect the current conditions, making a smaller time window more appropriate. As shown in Algorithm 2, the coherency assessment relies on the Euclidean distance between the event location indicated in the DENM and the vehicle positions specified in the corresponding CAMs.

Furthermore, the algorithm evaluates the similarity of the event provided in the DENM with those already received using a centroid-based approach, as outlined in Algorithm 3. This approach computes two centroids: a time centroid ($\tau_{centroid}$) and a distance centroid ($\delta_{centroid}$).These centroids are initialized with the detection time and position of the first DENM. Subsequent DENMs trigger updates to the centroids by averaging the old centroids with the detection time and position in the newly received message. The time centroid is used to create a time window, $|\tau_{centroid} - cadDen_{detTime}| \leq \tau_{eventType}$, within which a detection time is considered similar in temporal terms. From a spatial point of view, $\delta_{centroid}$ and a specified radius $\delta_{eventType}$ define an expected area for the event's position similarity assessment. This process produces a list of similar DENMs employed to assess the current DENM accuracy.

The resulting reputation score ($repScore$) is then applied to update the reputation of the vehicle responsible for generating the DENM.

Reputation Score Update. The edge node updates the reputation of the vehicles according to the following equation:

$$(r_{DID})_t = \alpha \times (r_{DID})_{t-1} + \beta \times ((r_{DID})_{t-1} + repScore) \tag{3.1}$$

where $(r_{DID})_t$ represents the updated reputation score of the source DID, while $(r_{DID})_{t-1}$ refers to the reputation score from the previous time step $t-1$. The parameters α and β define the respective contributions of past reputation and the new computed score. Their sum equals 1, ensuring a balanced contribution to the updated reputation score. The new reputation score combines the previous reputation and the score calculated via Algorithm 1. Reputation score is contingent on the type of misbehavior exhibited by the vehicle. Each misbehavior results in a degradation of the reputation score determined by the corresponding weight (e.g., ω_{msg} for message coherency). These weights are adaptable and can vary depending on the geographical area overseen by the edge node.

Reputation Score Storage. The final reputation score is stored in the Tangle through a zero-value transaction. For this to succeed, there must be no conflicts, such as two edge nodes updating

Table 3.1: Summary of ETSI Dataset Characteristics

Dataset Name	Description	Illustrative Features
DENM dataset	Comprises a subset of the information contained in DEN messages exchanged during the execution of the simulated scenario detailed in [79].	source; situation_eventType; detection_time; simulation_time; eventPos_lat; eventPos_long; eventPos_alt.
CAM dataset	Includes a selection of data extracted from CAMs exchanged within the context of the simulated scenario outlined in [79].	source; referencePositionLat; referencePositionLong; referencePositionAlt; simulationTime.

the same vehicle's reputation. In VANETs, it can be assumed that edge nodes are typically spaced far enough apart to prevent a vehicle from communicating with two nodes simultaneously. In real-world deployments, edge nodes are located either at the network edge, near the radio node (e.g., RSU), or at network aggregation points where they manage multiple radio nodes[47]. Hence, by defining the area managed by edge nodes as the coverage provided by their associated radio network nodes, the overlap between areas handled by different edge nodes becomes negligible. Furthermore, the time needed for a vehicle to move from one node's coverage to another exceeds the time required to update the vehicle's reputation.

Vehicles can request reputation data relevant to the area managed by a specific edge node. To access the most recent reputation scores, a vehicle must submit its VP. This ensures that only authorized users can retrieve this information and prevents malicious vehicles from accessing or influencing the reputation data.

3.1.4 ETSI-Compliant V2X Dataset

To evaluate the effectiveness of the DIVA system, a custom dataset[1] was generated, containing selected fields of V2X messages compliant with ETSI standard formats (section 2.1).

This dataset is based on the scenario described in [79], where the authors extended the Artery tool [81] to create dynamic VANET scenarios and present a DEN use case. Artery is a VANET simulator [82] integrated into the Objective Modular Network Testbed in C++ (OMNeT++) environment and incorporates SUMO [83] for road traffic modeling. It supports both safety and non-safety message formats defined by ETSI [20, 19] for VANET communications. Message exchanges were recorded using a custom OMNeT++ module, able to record all messages exchanged within the scenario.

The region of interest from [79] focuses on a highway junction, *Ingolstadt Nord*, in Ingolstadt, Germany. In this area, a vehicle performs an emergency stop at the intersection, creating

[1] https://github.com/MMw-Unibo/ETSI-V2V-Dataset

a collision risk and triggering the broadcasting of DENMs. Vehicles detecting such critical situations broadcast DENMs, which contain essential event data, such as *detection time*, *cause code*, and *event position*. These messages were included in the generated dataset, as they are crucial for detecting malicious behavior in the DIVA system.

Moreover, the custom OMNeT++ module collected CAMs generated within the same area, forming a crucial part of verifying the reliability of DENMs. This includes identifying any inconsistencies in detection time and vehicle positions. Table 3.1 provides an overview of the critical components of the DENM and CAM datasets with their relative features.

The dataset was further manipulated by introducing malicious instances into the DENM content. This involved injecting noise into critical fields, such as detection time, longitude, latitude, and altitude generated by sources randomly selected from the initial dataset. However, this might lead to nonsensical alterations for the respective columns. Hence, a normal distribution is created for each column, centered in zero, with a standard deviation tailored to match the specific deviation of that column. To ensure that the modified values remained realistic, independent distortions were applied even to messages from the same source, adding complexity to identifying malicious actors. This approach created a robust testbed for evaluating the DIVA system's ability to detect and handle misbehavior effectively.

3.1.5 Measuring DIVA Performance

This section explores DIVA threshold selection and its performance in detecting malicious messages by using the dataset presented previously and its manipulation. The algorithm has been implemented through a Python script[2] that processes every single message characterizing the DEN-based dataset. All experiments were conducted on a Linux virtual machine with 16 CPUs and 32 GB of RAM.

Threshold Selection

To enable DIVA to distinguish between malicious and non-malicious messages effectively, careful selection of the thresholds outlined in section 3.1.3 is essential. Thus, three distinct aggregation metrics, namely *mode*, *median*, and *mean*, are analyzed. These metrics measure the distances between the vehicle generating the event and the event location, derived by combining CAMs content with the information provided in the corresponding DENM.

The computation of these thresholds relies on the benign sample of the datasets. For message age, thresholds were taken from relevant reference documents (e.g., [84]). To determine which

[2] `https://github.com/MMw-Unibo/DIVA`

Table 3.2: Threshold Study with 20% of malicious vehicles.

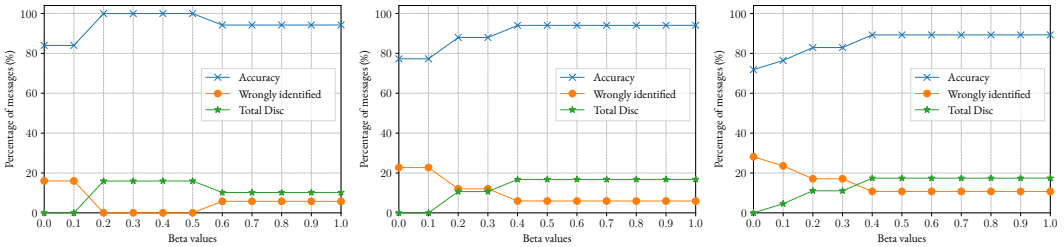
Threshold	Event Type	$\alpha = \beta$	TPR	TNR	FPR	FNR
Mode	dangerousEndOfQueue	0.5	100	49.51	50.49	0
	collisionRisk	0.5	87.23	92.8	7.20	12.77
	trafficCondition	0.5	100	78.48	21.52	0
	Total	0.5	99.93	78.14	21.86	0.07
Median	dangerousEndOfQueue	0.5	100	49.51	50.49	0
	collisionRisk	0.5	78.72	92.80	7.20	21.28
	trafficCondition	0.5	100	78.48	21.52	0
	Total	0.5	99.89	78.14	21.86	0.11
Mean	dangerousEndOfQueue	0.5	100	100	0	0
	collisionRisk	0.5	61.70	100	0	38.30
	trafficCondition	0.5	100	100	0	0
	Total	0.5	99.80	100	0	0.20

CAMs are considered for coherency computation (Algorithm 2), a predefined τ_{cam} value of 600 seconds was used, specifically chosen to address the scenario discussed in this section. However, as mentioned previously, the outlier detection algorithm allows flexibility in fine-tuning this value for different geographical regions.

To ensure consistent threshold evaluation, this analysis applied the same values for α and β (Equation 3.1). Table 3.2 shows the results obtained for each event type within the DENM dataset when α and β are both 0.5. It highlights the percentage of accurately classified messages, i.e., True Positive Rate (TPR) for malicious messages and True Negative Rate (TNR) for non-malicious ones. Misclassifications, including the False Positive Rate (FPR) and False Negative Rate (FNR) for malicious and non-malicious messages, are also shown. For brevity, metrics are reported for cases where 20% of the sources in the dataset are malicious. Most events fall under the *traffic condition* category, as it is the most commonly transmitted event in DENMs. Therefore, the event type *Total*, aggregating results regardless of the situation contained in the message, is notably influenced by their outcomes.

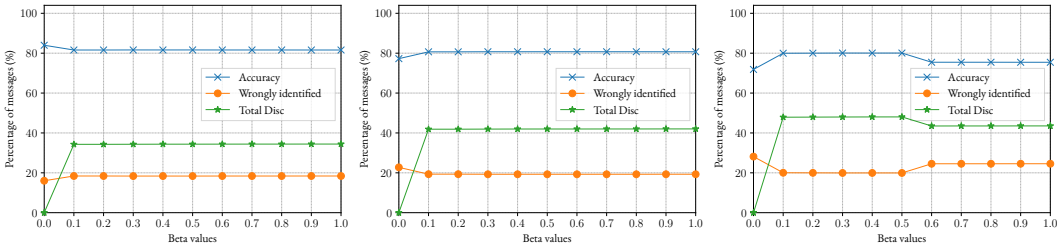
The table also demonstrates that using the *mode*, representing the most frequent distance value as $\delta_{eventType}$, results in the highest performance for detecting malicious messages, achieving a TPR of 99.93%. In contrast, the *median* threshold causes the algorithm to be overly conservative, leading to the rejection of approximately 20% of non-malicious messages. Meanwhile, using the *mean* ensures perfect identification of non-malicious messages, with a TNR of 100%, but leaves certain events, such as *collision risk* (TPR 61.70%), inadequately protected.

Based on this analysis, the subsequent section explores the performance of DIVA using both the *mean* and *mode* as $\delta_{eventType}$, while varying the values of α and β, as well as the proportion of malicious sources.



(a) 20% of malicious sources (b) 30% of malicious sources (c) 40% of malicious sources

Figure 3.3: Performance of DIVA using *mean* as threshold



(a) 20% of malicious sources (b) 30% of malicious sources (c) 40% of malicious sources

Figure 3.4: Performance of DIVA using *mode* as threshold

Performance Evaluation

Several experiments were conducted to determine the optimal combination of parameters α and β, with variations in the percentage of malicious sources within the DENM dataset. These experiments aimed to evaluate the performance of the DIVA system under different configurations.

Initially, DIVA was used to compute the reputation scores of sources from the manipulated versions of the DENM dataset. The dataset was then reprocessed to assess the reputation scores obtained in the previous step, with malicious messages identified as those generated by sources whose reputation score fell below a predefined threshold, $\tau_{repScore}$.

Figures 3.3 and 3.4 show the system's performance in identifying malicious and non-malicious vehicles, using *mean* and *mode* as thresholds. By comparing the graphs, DIVA performs better when using the *mean* as the threshold. This is particularly evident in Figure 3.3a, for values of β ranging from 0.2 to 0.6: the incidence of incorrectly identified messages approaches zero. For lower values of β, the performance of DIVA degrades as the influence of the new reputation score in Equation 3.1 is minimal. Indeed, the percentage of incorrectly identified messages increases to approximately 20% for these values of β in all the scenarios considered. On the other hand, when increasing the number of malicious sources, DIVA maintains a high accuracy for values of

β above 0.3, achieving around 94% with 30% of malicious sources (Fig. 3.3b), and 89% with 40% of malicious sources (Fig. 3.3c).

When *mode* is used as the spatial threshold, DIVA adopts a more conservative approach. While it accurately detects malicious messages, it also discards around 40% of the total received messages, including non-malicious ones. As shown in Figure 3.4c, this percentage rises to approximately 50% with the highest number of malicious sources.

These results demonstrate DIVA's effectiveness in identifying malicious sources, particularly with β values ranging from 0.4 to 0.6. By assigning them low reputation values, DIVA enables vehicles with benign intent to filter out these potentially harmful sources from V2X communications. This capability enhances the overall security and trustworthiness of the VANET environment, creating a safer network for all participating vehicles.

3.1.6 Security Evaluation of DIVA

To evaluate DIVA security, this section employs the Dolev-Yao adversarial model [85] and analyzes specific attacks related to VANET security [86, 87]. This analysis thoroughly assesses the system's resilience by assuming an attacker can intercept any message on the network, initiate communication with any entity, and even impersonate the intended recipient of any transmission.

Dolev-Yao adversarial model

Resistance to Eavesdropping Attack. In an eavesdropping attack, an adversary gains unauthorized access to sensitive information, such as location information, personal details, or messages exchanged between vehicles, by passively monitoring network traffic. It should be noted that V2V/V2I communications involve either public information (e.g., road conditions) or sensitive data, such as the location and direction of the vehicles (refer to Table 2.1). While listening to public information does not directly benefit potential attackers, in case of sensitive data transmission, the communication should be secured through end-to-end encryption using the receiver's public key. In DIVA, this key is shared directly in the DID Document. Along with vehicular information, a vehicle attaches its VP to prove the trustworthiness of the provided data. This approach mitigates the risk of adversaries intercepting valid VPs to transmit deceptive information, as DIVA relies on digital signatures requiring the attacker to have the sender's private key, making it nearly impossible to falsify data without detection.

Resistance to Replay Attack. Replay attacks occur when an adversary intercepts and retransmits previously recorded messages, creating misleading information that could lead to hazardous conditions. In the DIVA architecture, presented in section 3.1.3, edge nodes check message timestamps, discarding messages if they lack sufficient freshness to prevent outdated or reused data.

Furthermore, a unique random string or challenge is embedded in each VP, which protects against attempts to reuse the VP with a different verifier. As the VP's primary role is to verify the sender's authorization, any attacker without access to the sender's private key would find it nearly impossible to spoof the sender's identity or inject tampered data.

Resistance to Forgery Attack. In a forgery attack, a malicious entity attempts to impersonate legitimate vehicles or network entities to mislead nearby vehicles and manipulate their responses. In DIVA, each vehicle is assigned a unique DID linked to a cryptographic key pair. To inject malicious data into the system, the adversary would need to gain access to the private key of an authorized vehicle, which is highly unlikely given the system's reliance on secure, asymmetric cryptographic mechanisms.

Resistance to Sybil Attack. A Sybil attack is an extension of forgery, wherein an adversary creates multiple fake identities or vehicles to deceive network participants, disrupting routing protocols, manipulating traffic, or spreading misinformation. In DIVA, each vehicle is associated with a single, unique DID, meaning the adversary would need as many physical vehicles as identities to register multiple times. Purchasing multiple vehicles to compromise a cooperative awareness system is impractical for most attackers. Furthermore, once the reputations of the vehicles drop under a given threshold, other participants automatically discard all their messages. Therefore, in extreme cases where the adversary may own or be able to control many vehicles, its attack window is extremely narrowed to the time needed to go under the chosen threshold.

Vehicular Network-Oriented Attacks

Resistance to Global Positioning System (GPS) Spoofing Attack. In a GPS spoofing attack, an attacker transmits falsified location data to mislead vehicles. Despite DIVA cannot directly prevent this attack on an individual vehicle, it leverages messages locality to mitigate its impact. Specifically, DIVA reduces the reputation of a compromised vehicle based on inconsistencies detected in CAM messages from other vehicles. Once the reputation drops below a threshold, the vehicle is excluded from the VANET. For example, consider a fleet of five vehicles, two of which are under attack. By analyzing CAM messages from the three uncompromised vehicles, DIVA detects anomalies and lowers the reputation of the affected vehicles until they are removed from the network. This mechanism is further reinforced in environments equipped with RSUs, which can verify vehicle positions using cameras and radio sensing.

Resistance to Free-Riding Attacks. Free-riding attacks occur when malicious vehicles evade the authentication process while impersonating legitimate participants by embedding fake authentication tags obtained from neighboring vehicles. Within the VANET context, such attackers can exploit safety-related information without contributing to message forwarding, leading

to network fragmentation and degraded communication reliability. Moreover, in a reputation-based system, an attacker may leverage the reputation of other vehicles without establishing its own. In DIVA-based scenarios, a *free-rider* lacking authentication—without a valid DID—is automatically excluded from communications, as vehicles interact only with peers having a VP and a high reputation score. Furthermore, VPs includes nonces, ensuring uniqueness and preventing replay or identity reuse. As a result, *free-riders* cannot gain access to the reputation table since only vehicles with a valid VP are permitted to retrieve reputation data.

Resistance to Illusion Attacks. In VANET, illusion attacks exploit human psychological intuition through scene-aligned false message distribution [88]. Attackers create virtual traffic events by identifying the traffic condition and then broadcasting false warning messages. Unlike forgery attacks, which rely solely on false information, illusion attacks reinforce disinformation with real-world environmental signals, making them harder to detect. As mentioned in the context of forgery attacks, in DIVA, each vehicle is assigned a unique DID linked to a cryptographic key pair, ensuring authenticated communication within the VANET. Thus, an attacker must first gain access to the network to inject false messages. If the attacker gains access, DIVA can detect malicious activity by aggregating data from other vehicles in the vicinity or, if available, from RSUs. As a result, in DIVA-based scenarios, an illusion attack may only be effective for a limited time, after which the attacker's reputation score will fall below the threshold, leading to exclusion from the network.

3.2 V2V Platooning Quality Control: A TSN Slot-Based Scheduler

Despite the significant advancements in V2V communication protocols (see section 2.1), emerging applications such as remote and autonomous driving require more precise resource coordination and control among vehicles [89, 25, 90]. As mentioned in other sections, these scenarios heavily rely on broadcasting cooperative awareness and safety messages, often without acknowledgments [19, 20]. The loss of such messages could disrupt the correctness of some critical functionality by significantly increasing the risk of accidents. Moreover, since cooperative awareness messages are transmitted at fixed intervals, the likelihood of data loss rises, intensifying interference between nodes [25]. Addressing these challenges is crucial, necessitating the development of robust collision avoidance mechanisms specifically designed for vehicular communication environments.

To meet these demands, the next generation of V2V communication solutions will likely be driven by the integration of TSN [91, 92] in VANET. TSN comprises a suite of standards de-

signed to ensure deterministic Ethernet networks, particularly for real-time traffic. A fundamental element of TSN is time synchronization, accomplished through IEEE 802.1AS, an extension of the IEEE 1588 Precision Time Protocol (PTP) known as generic PTP (gPTP) [93]. This protocol relies on a grandmaster clock, typically a GPS receiver, to synchronize all devices within a gPTP domain. Clock Masters distribute time information to Clock Slaves, which adjust their clocks to maintain synchronized, cohesive communication. To ensure QoS, TSN employs traffic shaping techniques [91] like the Time-Aware Shaper (TAS), outlined in IEEE 802.1Qbv. TAS schedules frames for time-critical flows [94] using time-aware communication windows and cyclic time slots managed by a Gate Control List (GCL). This ensures low latency, minimal jitter, and high reliability, with PTP synchronization crucial for TAS's functionality.

In automotive Ethernet communications, TSN includes the IEEE 802.1DG profile, which ensures high reliability and low latency for in-vehicle networks. The profile prioritizes the timely delivery of crucial control and security messages. Thus, thanks to its characteristics, such as determinism, low latency, high reliability, and large bandwidth, TSN represents a promising technology for the next-generation in-vehicle networks [29]. This dissertation expands these TSN-based networks to enhance QoS in V2X communications. By integrating TSN with VANET as well as broader V2X frameworks [95], it is possible to exploit slotted protocols and time synchronization mechanisms to coordinate message transmission, reduce collisions, and improve overall communication reliability.

Hence, this section introduces a TSN-like Controller, TSNCtl, developed as part of the dissertation to enhance data delivery in vehicle platooning scenarios. TSNCtl orchestrates message dissemination over VANET by employing a Finite State Machine (FSM) to help in platoon formation and message dissemination for intra-platoon communications. Creating or joining a platoon is triggered by receiving ITS service-related messages or detecting proximity to an existing platoon. Once a vehicle becomes part of a platoon, TSNCtl intercepts each message generated by the ITS service running on the vehicle, determining the priority queue for its storage. Next, it propagates the messages based on the vehicle-associated time slots and the priority of each message. Furthermore, TSNCtl continuously adjusts slot allocation in response to nodes joining or leaving the platoon, maintaining stable and reliable communication despite the evolving topology.

This solution provides valuable insights into how TSN can be integrated into VANET communications to avoid message collisions and enhance overall QoS in dynamic vehicular environments. Additionally, this research demonstrates the feasibility of using a TSN-like approach to form and synchronize car platoons, thereby improving reliability and coordination in these complex systems.

3.2.1 Related Work

In multi-user communication networks such as VANET, users must contend for access to the shared communication medium. Traditionally, Carrier-Sensing Multiple Access Collision Avoidance (CSMA/CA) has been the go-to MAC protocol designed to reduce the probability of collisions in wireless communications. This protocol relies on nodes sensing the channel before transmitting data. Nevertheless, CSMA/CA has proven inadequate in meeting the stringent QoS requirements of VANETs, particularly in terms of latency and reliability [96].

This section discusses state-of-the-art approaches to improving data delivery and minimizing packet collisions in vehicular networks. It begins by examining early approaches based on random access techniques, then shifts to more advanced methods that leverage vehicle platooning to enhance network performance.

Random Access Techniques in VANET: Early Solutions

ALOHA is one of the earliest protocols designed to address collisions in multi-user networks, which operates on the principle that devices transmit data without checking whether the channel is busy. Any collisions require all stations to resend their data. Slotted-ALOHA improved pure ALOHA by introducing time slots within communications to enhance channel utilization. Reservation-ALOHA (R-ALOHA) further extended Slotted-ALOHA by adding a slot reservation mechanism that allows a device to temporarily "own" a slot after successfully using it. Borgonovo *et al.* [97] addressed the hidden terminal problem by proposing a Reliable Reservation ALOHA (RR-ALOHA) protocol, which adds additional information during message exchanges to reserve time slots and avoid collisions. The authors in [98] extended an improved version of RR-ALOHA [99], enhancing slot-reuse to increase the scalability in mobility-based scenarios.

More recent advancements in ALOHA-based MAC protocols include the works in [100, 101, 102]. Kim *et al.* [100] introduced a coordinate multichannel MAC providing contention-free broadcasting for safety messages, where vehicles reserve channels during the Service Channel Interval (SCHI) using Dynamic Frame-Slotted ALOHA, and the RSU coordinates the transmission order. The authors in [101] proposed an improved version of Slotted-ALOHA that disseminates access times over the Control Channel (CCH) period to reduce the risk of message collisions. In a recent work, Lai *et al.* [102] introduced a Framed Slotted Aloha (FSA) MAC protocol that leverages the capture effect, enabling successful access despite time slot conflicts to improve communication efficiency further.

Platooning-based VANET communications

Platooning represents a structured and cooperative approach to vehicular communication, where vehicles work together to achieve common goals, such as optimizing traffic flow and providing infotainment services [103, 104]. A platoon typically consists of a leader vehicle and several follower vehicles that drive closely together in coordination.

Many platoon-based proposals assume that the platoon leader allocates communication time slots to the followers, similar to Time Division Multiple Access (TDMA), defining specific time windows for communication activities [105, 106, 107]. For instance, in [105] Fernandes *et al.* provided a system that divides the control channel of the IEEE 802.11p into slots to ensure stable intra-platoon communications. Similarly, Segata *et al.* proposed a beacon dissemination strategy based on TDMA in [106] developed a TDMA-based beacon dissemination strategy, where the platoon leader initiates beacon transmissions followed by the other platoon members in a scheduled manner.

From a message collision avoidance perspective, Zang *et al.* [89] proposed a collision detection approach where transmitting nodes employ full-duplex channel sensing to detect simultaneous transmissions. In C-V2X, researchers are exploring methods to enhance scheduling algorithms, aiming to reduce or detect packet collisions in scenarios where vehicles are out of base station coverage like V2V [90, 25, 108].

Discussion

Despite these advancements, vehicular networks still face significant challenges related to scalability and reliability in dynamic environments. Many solutions focus on technology-specific approaches (e.g., WiFi or cellular networks), leaving space for technology-agnostic strategies that perform well regardless of the wireless technology. Moreover, many current proposals rely heavily on mathematical models and lack validation in large-scale, real-world scenarios.

The solution proposed in this section, TSNCtl, operates independently of the specific wireless technology utilized in the vehicle. It is designed to enhance the packet delivery ratio in intra-platoon V2V communications by establishing platoons, wherein each vehicle has a dedicated slot for message dissemination. Compared to ALOHA-based solutions, TSNCtl offers better scalability and is particularly suited for dense environments. Additionally, including priority queues ensures that critical messages are transmitted promptly, which is crucial for vehicular networks dealing with time-sensitive data.

Hence, this solution holds the potential to profoundly change the landscape of message dissemination within VANETs. For instance, combining cellular networks and TSN is seen as a promis-

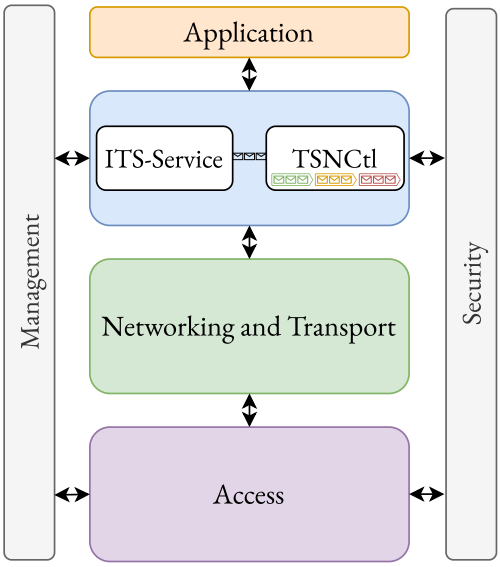


Figure 3.5: Integration of TSNCtl in ITS Stack

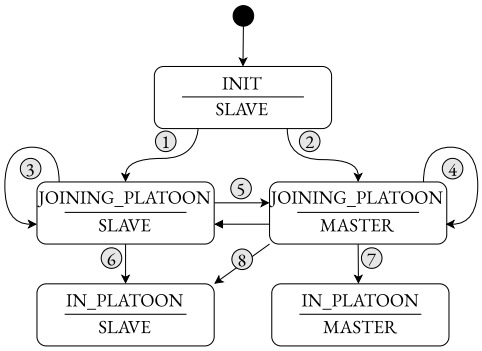


Figure 3.6: TSNCtl FSM Diagram

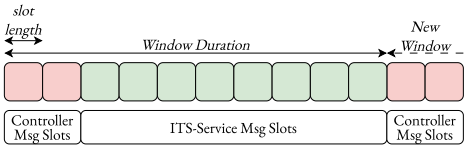


Figure 3.7: TSNCtl Slot-based Scheduling

ing path to support time-sensitive and low-latency applications, especially in conjunction with emerging 5G infrastructures [95]. Similarly, recent developments in IEEE 802.11 are working towards the full integration of TSN capabilities, enabling low-latency and ultra-reliable communications [92]. These advancements further highlight the growing importance of TSN in future vehicular communication systems.

3.2.2 TSNCtl for Time-Sensitive Applications in Vehicular Networks

As previously mentioned, the future of in-vehicle communications is increasingly focused on exploiting TSN to ensure low latency, reliability, and determinism for critical vehicular services [29]. The proposed solution assumes that TSN-enabled vehicles could leverage Radio and GPS technologies [109, 110, 111, 112] to synchronize themselves, facilitating synchronized and time-sensitive communications over wireless connectivity.

Figure 3.5 illustrates the integration of TSNCtl within the ETSI vehicular communications stack, introduced in section 2.1. TSNCtl operates as FSM that assists vehicles in platoon formation by determining which vehicles partake in network communications. The FSM initiates platoon creation or joining procedures upon receiving ITS service-related messages or detecting the proximity of an existing platoon. During this process, a platoon *master* is elected from among the vehicles, and this *master* is responsible for allocating communication slots to each platoon member. Vehicles utilize these slots to communicate at designated times, minimizing the risk of

collisions during communications with other vehicles within the platoon. In such a context, if a vehicle supports multiple safety-related applications, it may request additional slot allocations.

Once the platoon has been formed, TSNCtl intercepts messages generated by the ITS Service running on the vehicle and uses the pre-assigned communication slot to determine the optimal timing for message transmission across the network. To manage this process, TSNCtl maintains a series of queues, each assigned with a distinct priority level, wherein these messages are stored before transmission. Depending on the nature of the message, TSNCtl determines the appropriate queue for its placement and subsequent dispatching. Furthermore, TSNCtl deals with the dynamic topology of VANET environments by ensuring platoon stability as vehicles join or leave.

3.2.3 TSNCtl: FSM Approach to Enable TSN-like Slot-Based Scheduling

Platoon Formation and Initialization

As illustrated in Figure 3.6, the states that the FSM can assume are defined by a combination of node status, including *init*, *joining_platoon* and *in_platoon*, and a node role, *slave* and *master*, defining vehicle responsibilities within the platoon.

Initially, each vehicle configures parameters like communication window duration and slot length. The communication window establishes the interval for coordinated interactions among vehicles in the platoon. Slot length, on the other hand, specifies the timing for message transmissions, determining precise intervals when each vehicle can send data.

In the *init* state, the internal *TSNCtl* component remains idle until the specified window duration elapses. Referring to Figure 3.7, if the communication window duration is set to 100 ms (e.g., CAM), control-based messages are exchanged at multiples of this interval. The number of slots available per window depends on the slot length; for instance, a 100 ms window with 10 ms slots provides 10 slots.

Once the window elapses, the first two slots are designated for controlling operations, specifically to form or join a platoon (red slots in Figure 3.7). During the initial control slot, TSNCtl-related messages are sent, including node information and the message generation timestamp. In this phase, nodes compete for the *master* role by randomly generating messages within the slot's time frame to avoid collisions.

In the second control slot, TSNCtl processes the messages from the previous slot, transitioning to the *joining_platoon* state. As illustrated in Figure 3.6, in steps ① and ②, the node's role is determined by timestamps: the node with the earliest timestamp becomes the *master*, while the others take on the *slave* role. Within this slot, the *master* node dispatches a message containing details about slot allocations. These allocations can be determined using various policies, poten-

tially influenced by factors such as the number of slots requested by a node and the node type (e.g., emergency vehicle or standard car).

The *master* requires at least one nearby node to create a platoon. If no other node is detected, the *master* maintains the current status and restarts the procedure to form the platoon (step ④). A *master* node persisting in this state may become *slave* at the next controller slots(step ⑤). Conversely, in the presence of other nodes, its state transits in *in_platoon* (step ⑦).

Meanwhile, *slave* nodes await their slot allocation from the *master*, preparing to transmit data in their assigned slots. Once triggered, they confirm their role and transition to *in_platoon* (step ⑥). If no allocation is received, they restart the joining process in the next cycle (step ③).

In the *in_platoon* state, *slave* nodes transmit only during their designated slots. At the same time, *masters* also use the second control slot to handle communications, such as integrating new nodes into the platoon. For this integration, unaware of the platoon's presence, the latest node transmits a control message containing its timestamp during the first control slot. During the second control slot, the *master* responds with an updated slot allocation message, integrating the new node into the platoon's schedule.

Following platoon formation, each message generated by the ITS service is categorized based on priority and inserted into one of the available queues within the controller. The TSNCtl then disseminates these messages according to the allocated slots and priorities.

Platoon Stability and Dynamic Adaptation

To ensure platoon stability in dynamic environments where vehicles may join or leave the platoon unexpectedly, each *TSNCtl* instance monitors the successful transmission of messages within assigned time slots. If a vehicle fails to transmit within its allocated slot, it is excluded from the platoon, and its slot is reassigned. The reorganization procedure differs based on whether the departing vehicle is a slave or the master.

When a slave node leaves the platoon, the master handles slot reallocation by compacting the schedule during the next available "*first*" controller slot. It shifts all subsequent allocations to close the gap left by the departing node. The updated schedule is broadcast to the remaining nodes in the following controller slot, ensuring an efficient slot structure and preventing unnecessary communication delays.

If the node leaving the platoon is the master, different strategies are employed to maintain continuity. One approach is to trigger a new platoon formation, resetting the system and initiating a master re-election. Alternatively, the master role can be transferred directly to the next node in the current slot allocation (shift method), eliminating the need for re-election. The latter strategy avoids the needing a new master election, reducing message overhead and reorganization time.

This approach minimizes message overhead and reorganization time, ensuring a seamless transition while preserving the existing platoon structure.

Vehicles, however, may leave the platoon after successfully transmitting a message in their designated time slot. Their departure is only detected when they fail to participate in the subsequent time window. This delayed detection requires careful handling to prevent communication gaps and enable efficient resource reallocation.

TSNCtl primarily employs the shift method for master reassignment, ensuring robust and efficient platoon management even in highly dynamic scenarios.

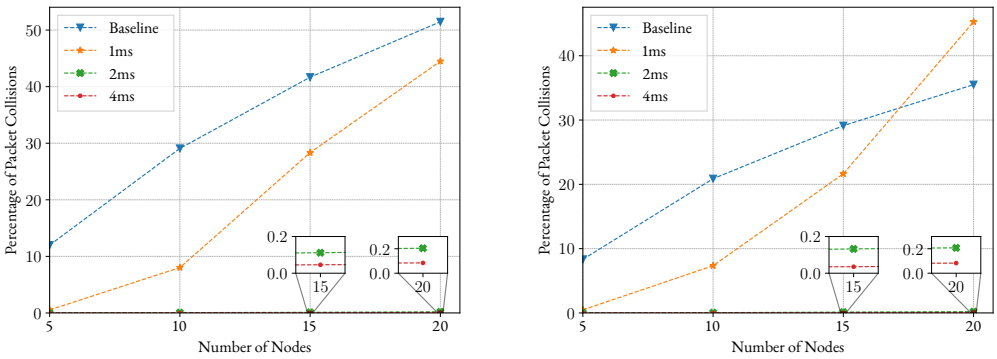
3.2.4 Measuring TSNCtl Performance

To evaluate the effectiveness of TSNCtl, the controller was developed using the OMNeT++ simulation framework. Specifically, INET 4.5, an OMNeT++ library for simulating network communications, was utilized to model ad-hoc V2V communications following the IEEE-802.11p standard. Two OMNeT++ simple modules were implemented: a mock service, which sends messages to the TSNCtl components (see Figure 3.5), and a mock application, which sends messages directly through the socket. Both modules generate UDP packets with a regular interval of 100 ms by following the CAM format defined by the ETSI standard. Packet sizes rely on the analysis conducted by the CAR2CAR Communication Consortium [113] in real-world scenarios. A *spawner* module was also introduced to generate and destroy vehicle modules in a defined area and at specified frequencies. The platoon formation area in this simulation corresponds to the V2V communication range (100 to 300 meters), with vehicle spawning intervals set to 1 ms or 100 μs.

Two sets of experiments were conducted to evaluate the feasibility of the proposed solution. The experiments examined the system's behavior under varying platoon sizes and vehicle entry frequencies, as well as the resilience of the solution under different packet sizes. The simulations were performed on a Linux virtual machine running OMNeT++ with 16 CPUs and 32 GB of RAM. The duration of each experiment was set to 1 minute simulation time, and each was repeated 10 times for statistical validity. This testing phase did not consider signal attenuation caused by obstacles such as roadside objects and foliage.

Packet Collision Analysis

Platoon Sizes. This set of experiments evaluated the impact of platoon size while maintaining a fixed packet size of 800 bytes [113, 114]. The communication window was set to $100ms$, and slot lengths of 1, 2, and 4 ms were analyzed. Figure 3.8 presents a comparison of intra-platoon packet collision rates between the CSMA/CA scenario (Baseline) and those using TSNCtl.



(a) Packet Collisions with a $100\mu s$ Vehicle Spawning Frequency

(b) Packet Collisions with a $1ms$ Vehicle Spawning Frequency

Figure 3.8: Collisions in Packet Transmission with Different Platoon Size

Figure 3.8a shows the percentage of packet collisions obtained when vehicles attempt to form a platoon with an interval of $100\mu s$ between their request, while Figure 3.8b with an interval of $1ms$. In both scenarios, *TSNCtl* correctly avoids packet collisions and helps in platoon formation with slot length above $1ms$. However, when the slot length is too short, TSNCtl's performance degrades significantly due to the increased likelihood of vehicle control messages being lost during the master election phase, thereby compromising platoon formation. This effect becomes particularly evident as the number of nodes increases: beyond 10 vehicles, TSNCtl performance degrades by reaching over 40% with 20 nodes in both scenarios analyzed in Figure 3.8. Conversely, when the slot length is set above $1ms$, TSNCtl achieves near-zero collision rates, ensuring robust and reliable platoon coordination. Indeed, in these cases the average percentage of collisions remains low, with values of 0.09% for a $2ms$ slot length and 0.035% for $4ms$ in the $100\mu s$ scenario (Figure 3.8a), maintaining similar performance in the $1ms$ scenario (Figure 3.8b).

This analysis highlights how slot lengths shorter than $2ms$ were insufficient for reliable message propagation within the platoon, leading to increased packet interference. Moreover, signal attenuation due to obstacles and foliage could exacerbate this issue —an aspect not yet explored in this study. These findings highlight the importance of selecting an appropriate slot length to optimize the performance of our solution.

Packet Sizes. In a second set of experiments, a platoon of 20 vehicles was generated at 1 ms intervals. As for previous experiments, window duration was fixed to 100 ms, while different slot lengths were analyzed. Figure 3.9 illustrates the percentage of collisions observed while varying packet dimensions in this scenario. The chosen packet dimensions for this analysis are derived from the data provided [113].

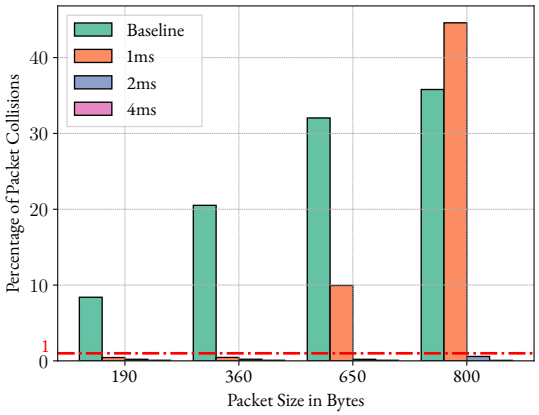


Figure 3.9: Collisions in Packet Transmission with Different Packet Size

The *Baseline* bar in the figure reveals a clear trend: collision rates increase as packet sizes grow, with an approximate 12% rise in collisions for each incremental increase in packet size. In contrast, when using the TSNCtl controller, the collision rate remains consistently below 1% for slot lengths above $1ms$. It should be noted that even with a slot length of $1ms$ and small packet sizes (< 650 bytes), TSNCtl effectively maintains a collision rate below 1%. This suggests that packet size is crucial in determining the optimal slot length. Hence, the potential for the platoon master to dynamically adjust the slot length based on real-time network conditions is evident; however, this aspect has not been analyzed in this study.

4 A Novel Design for Vehicular Computing in 5G Deployments

The VC paradigm, introduced in this dissertation, defines a cloud continuum of resources by integrating resources distributed across cloud, edge, and far-edge layers (Figure 1.1). This Chapter extensively studies the challenges involved in realizing the VC paradigm, including managing resource volatility, creating adaptable and dynamic environments, and ensuring the reproducibility of experimental results.

A novel architectural design is provided within the context of 5G networks, leveraging the ETSI MEC reference framework. This design integrates stationary far-edge/vehicular resources into the cloud continuum, ensuring low-latency, high-bandwidth connectivity. In addition, this Chapter details the implementation of a simulation tool that replicates this environment, providing a standardized platform for researchers and engineers where they can design and test their algorithms and applications while exploiting at the same time a standard enabled VC paradigm.

4.1 Virtualizing Vehicular Resources to Empower MEC Nodes

Edge computing ensures optimal performance for latency-sensitive 5G applications such as metaverse and real-time control systems. Beyond its impact on entertainment and responsive control, edge computing also supports applications critical for safety and security, particularly in ITS and the landscape of fully integrated smart cities [115]. As the digital infrastructure of cities evolves, the convergence of edge and cloud environments is set to become even more integral to modern urban ecosystems.

New opportunities and challenges emerge as many businesses leverage this shared edge-cloud environment. Unlike traditional cloud deployments in data centers, edge infrastructures face significant resource limitations and may struggle to meet specific applications' resource demands and associated QoS requirements [16]. Moreover, the technical challenges associated with advanced edge infrastructures are exacerbated by the convergence trends, which aim to provide end-users

with seamless access to a wide range of services, regardless of where these services are hosted within the edge-cloud continuum. For these reasons, the need has emerged to identify new resources that can support the edge infrastructure dynamically, thus enabling service availability in dense and congested deployment scenarios.

This dissertation proposes an extension of the ETSI MEC standard, introduced in section 2.3, to leverage resources at the far-edge layer, exposed and made available in a standardized way. The extension defines for each MEC-H (edge node) an Area of Interest (AoI) within which far-edge node resources are collected. Once these resources join the MEC-H resources pool, they are available for allocating MEC-compliant applications. This design allows the handling of node mobility by utilizing MEC standard services, thereby supporting the application migration when nodes leave the AoI.

This section provides design details and technical examples of how vehicles can fit into this architecture by contributing with their resources. The proposed architecture marks the development of a VC-enabled solution that supports multi-vendor and multi-operator scenarios, leveraging a well-established standard to ensure interoperability and scalability.

4.1.1 Related Work

Extensive research has been dedicated to exploring the use of vehicular resources to enhance service delivery at the network edge [4, 3]. Opportunistic vehicular resources can be leveraged for various tasks to support the increasing demands of applications in vehicular networks. For instance, vehicles can function as relay nodes [116, 117, 118] improving network connectivity, or as computing nodes [119, 120, 26, 118, 27], reducing the impact on edge node performance.

In centralized approaches, Huang *et al.* [120] proposed an architecture where the central node at the network edge receives task requests and distributes them as sub-tasks to selected parked vehicles. To utilize vehicles as part of the storage and networking infrastructure while parked, Dressler *et al.* [118] suggested augmenting RSUs capacity with the resources of parked vehicles.

Other works focused on dynamically forming micro-datacenters using in-vehicle resources without infrastructure requirements [119, 121, 122]. For example, Kamakshi *et al.*[121] considered vehicles' relative mobility (i.e., relative speed and distance) to aggregate vehicles in communities, while the authors in [122] designed an algorithm using the fuzzy logic for vehicular cluster formation. Feng *et al.* [119] presented a workflow for the autonomous formation of groups of vehicles, utilizing an algorithm based on ant colony optimization to schedule job distribution. Similarly, the authors in [27] introduced a decentralized approach, offloading task execution to parked vehicle resources available nearby.

In these environments, it is crucial to incentivize vehicle owners to contribute their resources. To tackle this challenge, Li *et al.* [123] defined a contract-based incentive mechanism to persuade vehicle owners to rent out their resources. Similarly, the authors in [117] propose an auction-based model where participating nodes compete to lend their resources in an extended vehicular resource pool.

Discussion

Due to the inherent mobility of vehicles, the capacity of the vehicular resource pool is subject to continuous fluctuations, making task offloading decisions more complex. The necessity for this might also arise due to an inaccurate estimate of residual resource availability. In this context, most of the works propose algorithmic strategies used to evaluate the probability of nodes to complete task execution [120, 26, 27, 124, 28]. These approaches are probabilistic, neglecting practical considerations such as nodes refusing to partake in the resource pool or leaving during task execution. Furthermore, existing proposals do not consider the complications of multi-vendor and multi-domain environments.

The architecture proposed in this section builds on the ETSI MEC standard, incorporating far-edge resources provided by parked vehicles. By leveraging standardized ETSI MEC interfaces, vehicle resources can be transparently accessed and made available for task offloading. The architecture integrates mechanisms to deploy and distribute applications across the resource pool while addressing resource volatility through a transparent migration process that utilizes existing constructs. Thus, it considers the dynamic nature of these resources, such as vehicles leaving the parking lot while executing tasks.

As an extension of the ETSI MEC standard, this architecture addresses the earlier challenges by enabling better integration with cloud resources and facilitating interoperability across heterogeneous technologies. This provides a more robust solution for managing vehicular resources in dynamic, multi-vendor, and multi-domain environments.

4.1.2 Extending ETSI MEC to Dynamically Exploit Vehicle Resources

As illustrated in Figure 4.1, the proposal turns the MEC-H into a logical entity that can dynamically include and release computational resources from multiple VIs. The architecture enables the deployment of applications on MEC-H (local) and vehicular/far-edge node (remote) resources, all while actively addressing concerns related to resource volatility.

Each MEC-H defines an AoI (dashed circle in Figure 4.1) where remote resources are collected and integrated into its resource pool. The AoI can overlap with the coverage of one or multiple base stations, depending on the MEC-H's location, whether at the network edge (close to base

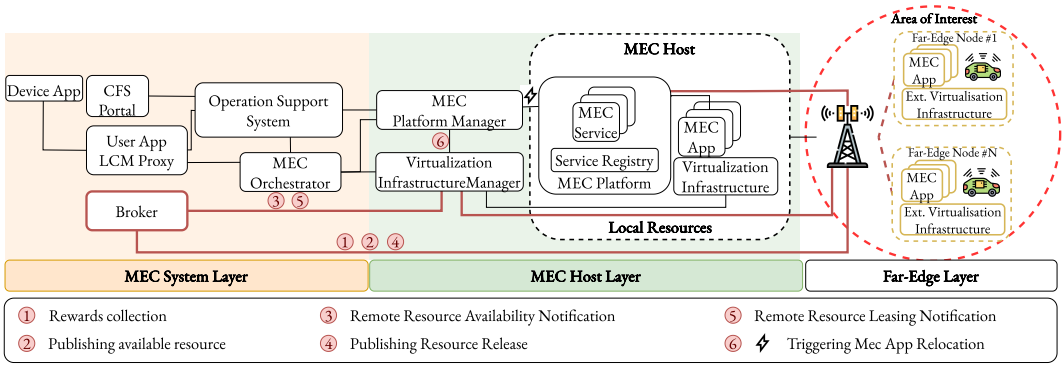


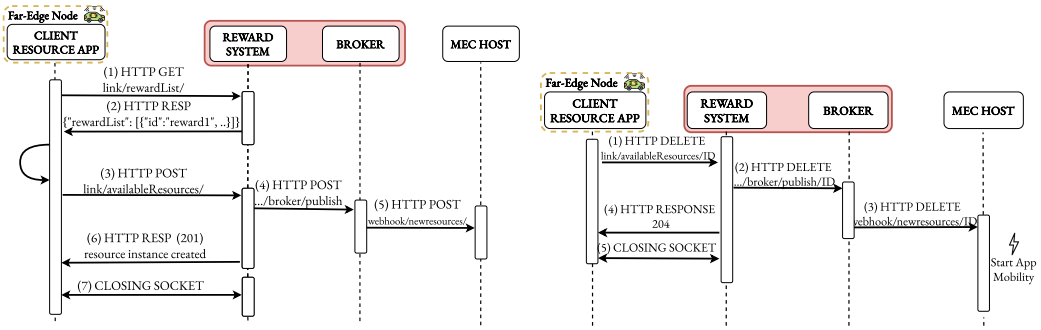
Figure 4.1: Proposal: Extending MEC Host Resource Pool

stations) or in central network aggregation points [47]. In the scenario considered in this dissertation, the AoI aligns with a parking area where the MEC-H operates.

The extended MEC architecture introduces a Broker, an external entity running at the MEC system level to model the resource acquisition procedure. The Broker enables a publish-subscribe system, allowing MEC-H subscriptions to the AoI and managing their notification whenever a new vehicle enters or leaves the area. The links and interactions within this extended architecture are highlighted in Figure 4.1, highlighting the Broker's role in coordinating these processes.

Upon entering the AoI, vehicles use a device-initiated scheme to request available rewards contextualized to the AoI (step ①). When a vehicle agrees to participate, it publishes the resources it is willing to contribute (step ②). Conversely, when a vehicle leaves, the MEC-H is notified, and the vehicle's resources are removed from the pool (step ⑤). Next, the mobility procedure, which extends the AMS API provided by the ETSI standard, takes place (step ⑥ Figure 4.1), ensuring seamless intra-host migration from a vehicle leaving the parking area to the local resources of the MEC-H. However, if the vehicle does not provide a departure notification, the broker cannot detect the change autonomously. In such cases, the client running the application on the departing vehicle will identify the unresponsiveness and trigger re-instantiation, indirectly signaling the vehicle's exit (step ④).

Resource allocation is primarily managed by the VIM component of the ETSI MEC architecture, which oversees the MEC-H resources (section 2.3). The VIM handles a heterogeneous pool of distributed resources and is aware of the single contributions that each host brings in terms of capacity. Once registered to the MEC-O, the VIM specifies the content of interest to the Broker corresponding to the AoI parameters (e.g., circle center and diameter). Upon new resource acquisition, the VIM records the endpoint and capacity information, making these resources available for MEC-compliant application deployment.



(a) Resource Acquisition Sequence Diagram (b) Resource Releasing Sequence Diagram

Figure 4.2: Sequence Diagram Device-initiated Scheme

The following sections provide a detailed explanation of the remote resource management scheme, emphasizing essential interactions within this framework.

4.1.3 Remote Resource Management in Extended MEC Nodes

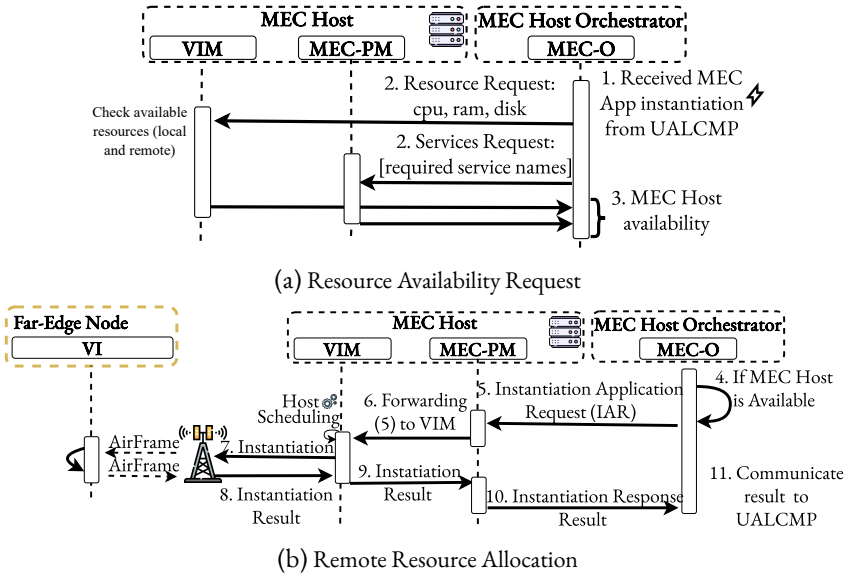
This subsection outlines the sequence of operations involved in acquiring, allocating, and releasing dynamic resources in extended MEC-Hs. Specifically, the following sequence diagrams illustrate the interactions between vehicles and the MEC resource pool, focusing on the resource acquisition, allocation, and release lifecycle as vehicles enter or leave the AoI.

Resource Acquisition

Figure 4.2a outlines the steps involved in the process of a vehicle joining the MEC-H resource pool as it enters the AoI. The initial phase employs an incentive-based approach, where vehicles receive a set of available rewards for contributing their resources. After evaluating the available options, the vehicle selects its preferred reward, signaling its willingness to contribute resources to the pool (steps (1) - (3)). In the second phase, the interaction shifts towards the resource Broker, enabling the integration of new resources. The Broker gathers the vehicle's resource details and communicates them to the MEC-H associated with the AoI (steps (3) - (6)). This ensures the MEC-H is aware of the newly available resources and can manage them accordingly.

Resource Allocation

Figure 4.3 demonstrates the interaction flow in instantiating an application on remote resources. As indicated by the standard (section 2.3), the process is initiated by the Device App, which sends a request for a new MEC App instantiation. This request is forwarded through a service chain from the UALCMP up to the VIM (steps (1) - (6) in Figures 4.3a and 4.3b), where the actual



(a) Resource Availability Request

(b) Remote Resource Allocation

Figure 4.3: Sequence Diagram Remote Resource Allocation

allocation of the app is performed. The VIM manages all the information related to the vehicles within the pool, maintaining a comprehensive dataset for each vehicle to facilitate efficient resource scheduling.

Based on the scheduling algorithm, app requirements, and the available contributions from each vehicle, the VIM selects the most appropriate remote host for deploying the application (step (7)), finally relaying the selected node information back to the UALCMP.

Resource Leaving

Figure 4.2b shows the interactions needed when devices leave the resource pool. Upon receiving a departure notification, the MEC-H removes the concerned resources from those available in the pool. This triggers the mobility management process for applications running on the departing vehicle, ensuring they are either gracefully terminated or migrated to other available hosts in the pool. The details of this process are outlined in section 4.1.4.

4.1.4 Migrating MEC Apps in Extended MEC Nodes

The version of AMS included in this architecture supports intra-host migration for applications running on nodes leaving the resource pool, ensuring service continuity transparently to the client actors. This is achieved by extending the AMS [125] API provided by the standard, structured as a publish-subscribe system that coordinates migration events across all related modules. Each

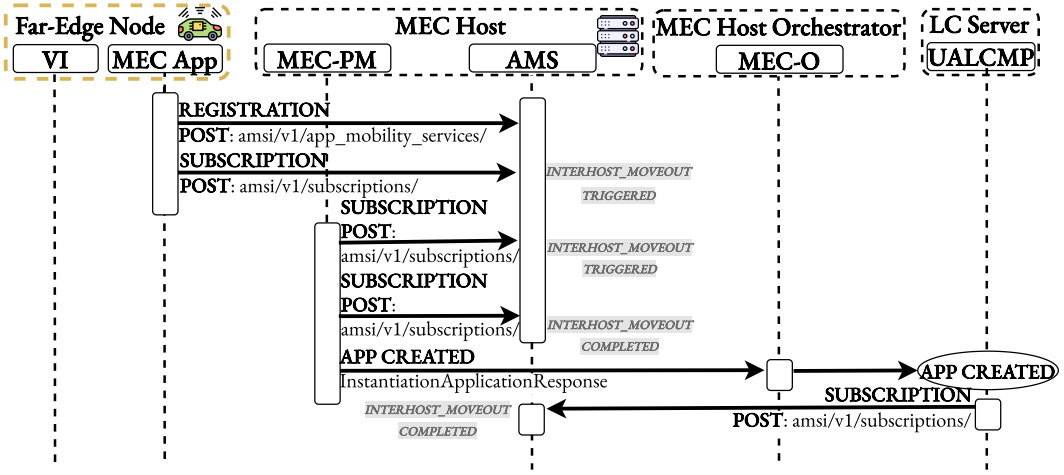


Figure 4.4: AMS Interaction Schema - Subscription Process

migration procedure outlines the specific operations that MEC Apps must perform to support these scenarios effectively.

Subscription Process

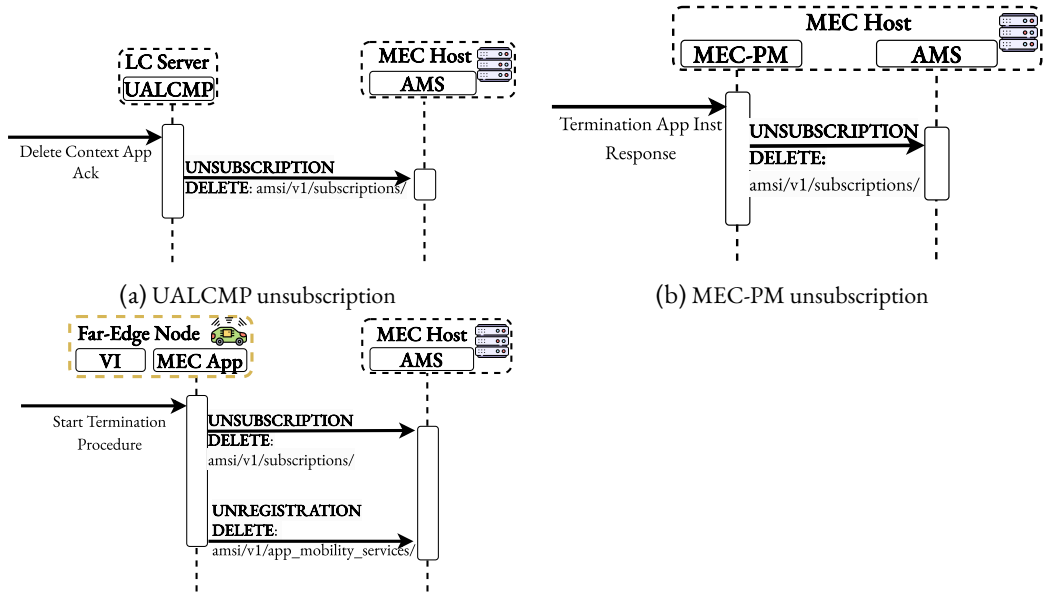
To enable migration, it is essential for the AMS to systematically recognize and manage each new MEC App. Figure 4.4 delineates the operations executed by involved MEC components upon the instantiation of a new MEC App.

Upon startup, the MEC App registers with the AMS and receives a unique registration identifier. Next, it subscribes to the `INTERHOST_MOVEOUT_TRIGGERED` event to ensure it is notified when migration is initiated, triggering the necessary context transfer process.

The MEC-PM subscribes to `INTERHOST_MOVEOUT_TRIGGERED` and `INTERHOST_MOVEOUT_COMPLETED` events to receive notifications when a vehicle/far-edge node leaves the MEC-H resource pool and when migration is completed. These notifications are necessary to start the migration and update the app information. Similarly, the UALCMP subscribes to the `INTERHOST_MOVEOUT_COMPLETED` event to collect new MEC App location, ensuring to update the requester user accordingly.

Unsubscription Process

When a MEC App is deleted or migrated, the relevant MEC components must unsubscribe from associated events, as shown in Figure 4.5. The UALCMP, MEC-PM, and related MEC App unsubscribe from all `INTERHOST_MOVEOUT` events. Once unsubscribed, the application is properly



(a) UALCMP unsubscription

(b) MEC-PM unsubscription

(c) MEC App unsubscription and unregistration

Figure 4.5: AMS Interaction Schema - Unsubscription Process

unregistered and removed from the system, ensuring resource cleanup and preventing redundant notifications.

Migration Context Trigger

Figure 4.6 (steps A and B) describes the interactions triggered when the migration process starts. This trigger prompts the instantiation of a replica of the migrating MEC App within the central MEC infrastructure, i.e., MEC-H handling the remote resources. It should be noted that this process is designed to allow future enhancements, such as selecting an alternative host based on scheduling algorithms.

When a participating vehicle/far-edge node leaves the AoI, the Broker notifies the VIM, which triggers the resource-release process (Figure 4.2b). At the same time, the VIM checks for any MEC Apps running on the leaving remote host and triggers migration as needed.

The `ParkMigrationTrigger` message, generated by the VIM when MEC Apps are running on a node preparing to exit the resource pool, is sent to the MEC-PM. This, in turn, forwards the `INTERHOST_MOVEOUT_TRIGGERED` event to the AMS (Figure 4.6 A). The AMS then informs all subscribed modules about the event. The MEC-PM sends a `ServiceMobilityRequest` to the VIM, requesting that the MEC application be moved to another host. While this may seem redundant, as the VIM receives the notification twice, it remains necessary. In typical migrations between dif-

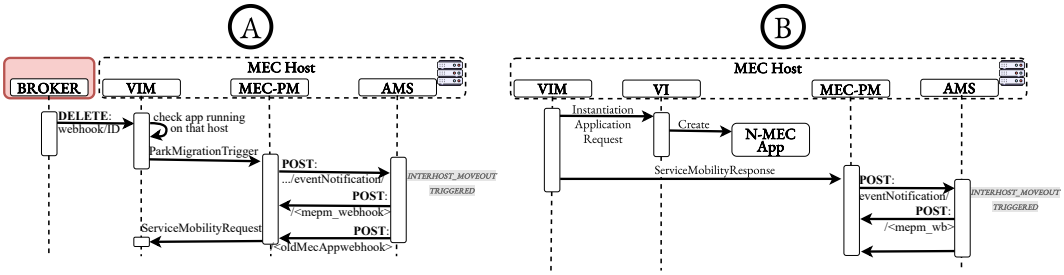


Figure 4.6: AMS Interaction Schema - Migraion Context Trigger

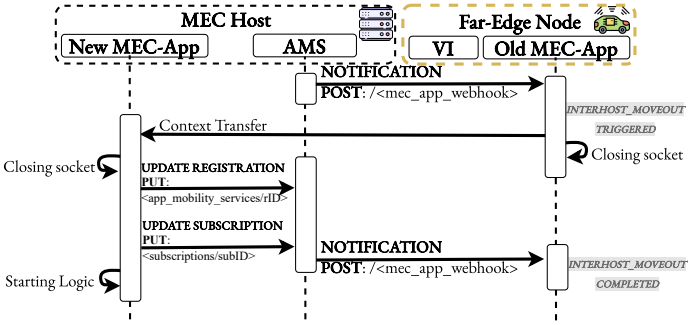


Figure 4.7: AMS Interaction Schema - Migration Context Transfer

ferent MEC-Hs, the `ParkMigrationTrigger` is not used, meaning the VIM would not receive the event twice.

As shown in Figure 4.6 step B, in the final step MEC-PM confirms the successful instantiation of the application on a new host and issues a new `INTERHOST_MOVEOUT_TRIGGERED` notification. This updated notification includes the location of the newly deployed application, making it available to all relevant system actors.

Migration Context Transfer

The context transfer procedure ensures synchronization between the migrating MEC application and its newly instantiated counterpart, minimizing service disruption for client applications.

After initialization, the new MEC App registers with the AMS and waits for the migrating MEC App to transfer the operational state. This is triggered by the `INTERHOST_MOVEOUT_TRIGGERED` notification on the migrating app, including all the information related to the location of the new instance (Figure 4.7), as mentioned earlier in this section.

Figure4.7 illustrates an example of context transfer of a MEC App migrating from the far-edge node to the MEC-H resources. The migrating MEC App establishes a communication channel with the new instance to transfer the necessary service state. Once the state is moved, the new

MEC App notifies the AMS, which then updates the UALCMP with the new location. The final step involves propagating the new location through the Device App to the user application, ensuring it is fully aware of the new service instance. The UALCMP receives a notification from the AMS, triggered by the `INTERHOST_MOVEOUT_COMPLETED` event, to complete this process.

4.2 Simulating and Validating Vehicular Computing Applications

As discussed earlier, most research in vehicular computing does not adhere to a standard for integrating mobile resources. Much of the existing work also focuses on algorithmic strategies that calculate the probability of task completion for each mobile node before task allocation, often overlooking migration. This section introduces a simulation model that implements the extended ETSI MEC standard described previously. The model is built on the OMNeT++ network simulator, incorporating the Simu5G library to model the 5G network and communication components.

OMNeT++ [126] is a widely used discrete-event simulation framework that models various network types and communications. It uses modules as its core elements, which exchange messages through gates and connections. Module behavior is defined in C++, while their structure, including gates and parameters, is described using Network Description Language (NED). On top of OMNeT++, Simu5G[127] models both the core network and the RAN of a 5G network by implementing 3GPP-compliant protocols and a customizable physical transmission system. It also supports advanced 5G network features like handover and inter-cell interference coordination, allowing simulations that include heterogeneous gNB base stations.

The simulation model presented in this section implements the architecture introduced in the previous section, specifically considering its deployment in a 5G environment (refer to section 2.3). This model improves the ETSI MEC standard by incorporating resources from parked vehicles within an AoI into the edge resource pool, making them accessible through standardized interfaces. The modeled architecture allows dealing with some of the primary challenges arising in vehicular computing environments, such as integrating cloud resources and enabling the coexistence of heterogeneous technologies. It also tackles resource volatility—nodes dynamically joining or leaving during service provisioning—through a migration mechanism compliant with the ETSI standard (see section 4.1.4).

This section highlights the solution's contribution to advancing the current state-of-the-art. It then introduces simulation model modules designed to create the extended ETSI MEC architecture presented in section 4.1, and it validates the feasibility of this architecture by providing

Table 4.1: Difference Between Existing Solutions and Proposal

Work	Comm. Model	Mobility Control	Resources	Standard Compliant	Open Source	Infrastructure support
Proposal	5G SA and mode 1	AMS API	Relatively Static	√	√	Needed
[128]	802.11p	N.A.	Static	X	X	Needed
[123]	N.A.	N.A.	Static	X	X	Needed
[27]	N.A.	N.A.	Static	X	X	Needed
[118]	802.11p	N.A.	Static	X	X	Not Needed
[28]	N.A.	N.A.	Static and Mobile	X	X	Needed
[124]	802.11p	N.A.	Static and Mobile	X	X	Not Needed
[119]	802.11p	N.A.	Static and Mobile	X	X	Not Needed
[122]	LTE mode 4	N.A.	Mobile	X	X	Not Needed
[129]	N.A.	N.A.	Mobile	X	X	Not Needed
[130]	5G (N.A.)	Custom	Static and Mobile	X	√	Needed
[131]	N.A.	N.A.	Static and Mobile	X	√	Needed

extensive experiments on the simulation model. Furthermore, it offers a practical use case to illustrate how this simulation tool can support researchers and engineers in developing and evaluating innovative algorithms, applications, and protocols within the vehicular computing domain.

The simulation framework is publicly accessible to the researchers through the GitHub repository[1].

4.2.1 Related Work

Table 4.1 summarizes the key characteristics of existing solutions that provide a Vehicular Computing model. The Table highlights various aspects, such as the communication model considered during experimentation, how each solution handles the high dynamism of VANET environments, the types of resources used to create dynamic computing nodes (i.e., mobile or static), adherence to standardized edge computing documents, the provision of an open-source platform for designing and testing other proposals, and whether infrastructure support is required to exploit vehicular resources.

In [128], the authors proposed the Vehicular Static Cloud-VANET model, which aims to establish a cohesive infrastructure for hosting ITS services. This model uses a centralized controller to integrate resources from parked vehicles equipped with IEEE 802.11p-enabled OBU with cloud resources and manages requests from on-road vehicles. Job assignment is based on vehicle residency time, i.e., the duration a vehicle remains part of the vehicular cloud, avoiding the need for task migration when a vehicle leaves the parking lot. Similarly, in [123], the authors use residency time for task allocation on parked vehicles, defining predefined contracts to incentivize vehicle owners to provide in-vehicle resources. Also focusing on parked nodes, the authors in [27] in-

[1] https://github.com/aferaudo/Simu5G/tree/feat/vim-extension

troduced a parking edge computing paradigm that pools computational resources from parked vehicles in urban areas to assist edge servers in offloading tasks. They presented a task scheduling algorithm and trajectory prediction model, showing improved offloading performance in urban environments. Dressler *et al.* [118] extended an existing routing algorithm to dynamically create distributed storage and improve network connectivity by exploiting resources from vehicles as they enter and leave parking lots without requiring RSUs. Extensive simulations were conducted using OMNeT++ and SUMO [83].

In [124], the authors presented an autonomous vehicle cloud formation algorithm, utilizing a predictive metric called companion time—the period vehicles stay in proximity to each other—to enable the formation of vehicle clouds in parking scenarios and in slow-moving traffic. Similarly, Fan *et al.* [28] proposed a joint task offloading and resource allocation strategy for clouds of vehicles covered by a base station. Feng *et al.* [119] designed a framework that uses beaconing to form vehicle clouds dynamically. Bute *et al.* [122] explored a solution that sees vehicles dynamically forming clusters on highways to create a collaborative computing platform with cloud infrastructure. The cluster head is selected using a fuzzy logic algorithm that ensures stable connectivity between nodes, supporting reliable communication. Hayawi *et al.* [129] explored a similar scenario, using Unmanned Aerial Vehicles (UAVs) as computing nodes in the solution named Skywalker, where UAVs provide radio coverage in dead zones.

Discussion

The solutions proposed in the above works do not adhere to a standard for integrating vehicular resources into the cloud continuum. Instead, most of them introduce new architectures or workflows for acquiring in-vehicle resources. Additionally, as outlined in Table 4.1, *Mobility Control* column, most works propose algorithmic strategies to evaluate the probability of nodes completing task execution. However, they do not deal with application migration when a node executing tasks leaves the cloud. Although the authors in [132] formalize the service migration process for parked vehicles as they leave the parking lot or refuse to continue providing services, it primarily focuses on a mathematical formulation of the migration problem, overlooking real-world challenges in VC. These include procedures for vehicle resources, such as join and leave procedures, incentive mechanisms to motivate vehicle owners and standardization.

One solution supporting migration in this context is VFogSim, proposed by Akgül et al.[130]. This open-source simulator adopts a data-driven approach, relying on real-world input data for accurate modeling. However, obtaining such data can be challenging, potentially limiting its usability in specific research scenarios. Additionally, the simulator supports only V2I-based communications and lacks an incentive mechanism for vehicle owners. Similarly, Wei et al.[131] introduced

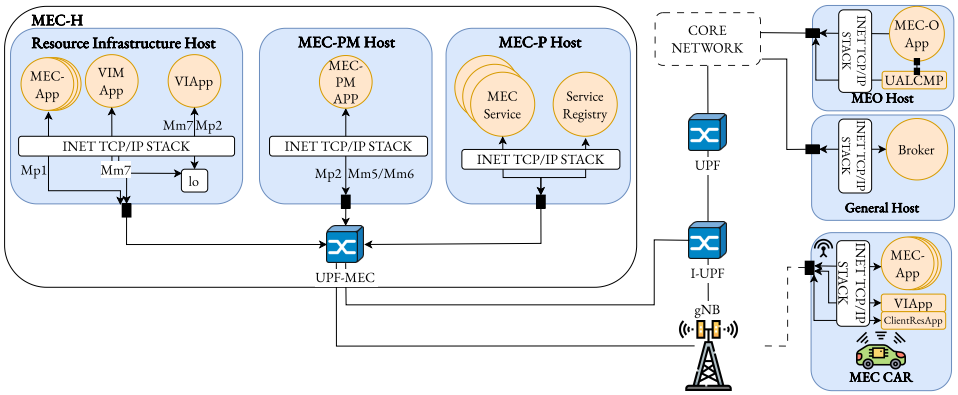


Figure 4.8: Simulation Tool Modules Structure

AirFogSim, a modular simulation platform enabling computation offloading in UAV-integrated VFC.

Current state-of-the-art simulation tools lack a comprehensive vehicular computing platform that enables researchers to design and test their algorithms and applications within a standard-compliant environment. The simulation tool presented in this dissertation fills this gap by providing a versatile platform for running simulations with either generated or real-world input data, producing meaningful results.

Moreover, compared to other solutions, this simulation tool incorporates a MEC-assisted migration procedure for tasks assigned to vehicles that either leave the parking lot or decline to continue providing services (see section 4.1.4). Currently, the tool supports quasi-mobile vehicles, i.e., those exiting the parking lot. Still, by supporting ETSI-compliant APIs for service migration, the platform is also adaptable to acquiring resources from moving vehicles.

4.2.2 Simulation Model: OMNeT++ Modules

Figure 4.8 illustrates the structure and deployment of the main components of the simulation model. From a structural perspective, the model includes a set of physical machines that host the key entities of the extended MEC architecture, developed as applications. This design allows each MEC component to be abstracted from the machine on which it operates.

The core of the MEC-H consists of components, including the Resource Infrastructure Host, the MEC-PM, and the MEC-P, which manage the life cycle of MEC Apps and provide MEC services in compliance with established specifications.

As mentioned earlier, the proposed architecture leverages vehicle resources to enable the execution of MEC-compliant applications at the far-edge layer. To enable this, the simulation model defines the MEC car module, which extends the NR UE (NRUE) defined in Simu5G. This mod-

ule wraps any 5G-enabled device and provides computational resources (e.g., CPU, RAM, and storage) necessary to run applications orchestrated by the MEC-H. The capabilities of this module are two-fold. First, it runs a *ClientResApp* initiating joining procedures to the resource pool of a specific MEC-H, receives a list of rewards, and selects one based on whether the vehicle can join. At the same time, this application manages resource release procedures when the vehicle exits the AoI(see section 4.1.3). Second, it executes the VI application, which manages local resources according to instructions from the central MEC-H and is responsible for deploying or deleting any MEC Apps. The VI module is designed to execute on any host with resource infrastructure, enabling the integration of any 5G-enabled device into the MEC-H resource pool. This could be a daemon that virtualizes and makes the device's resources available in an actual deployment.

In this extended architecture, MEC-Hs can support local and remote resources. Consequently, the VIM module must handle the scheduling, preparation, and release of both local and remote resources (see section 4.1.3). Scheduling is accomplished through an extensible algorithm system that selects the optimal host for application deployment based on specific semantics that may favor certain behaviors. Once a host is chosen, the VIM interacts with the remote hosts through the VI deployed on top of them to handle remote commands for allocating, relocating, and terminating MEC Apps.

Given the characteristics of the extended MEC architecture proposed in this dissertation, a vehicle may leave the resource pool at any moment, potentially causing service disruption for the applications it hosts. To mitigate this risk, a migration service is essential to ensure service availability and minimize delays. The simulation model addresses this requirement by custom implementing the AMS, described in section 4.1.4. This version extends the standard version to support MEC-assisted intra-host migrations and address volatility issues transparently. Specifically, each MEC App can be relocated from a remote host to the central infrastructure while also managing context synchronization between the two applications.

4.2.3 Measuring Extended MEC Nodes Performance through a Simulation Model

The simulation model presented in this section serves two primary purposes: first, it is designed to validate the architecture described in section 4.1, and second, it provides a platform for researchers and engineers to test and develop applications leveraging the vehicular computing paradigm. This section presents the performance evaluation of the simulation model. The first analysis compares network-induced delays across three service delivery modes to validate the model. Following that, resource management performance is evaluated using real-world data from a parking lot in Arn-

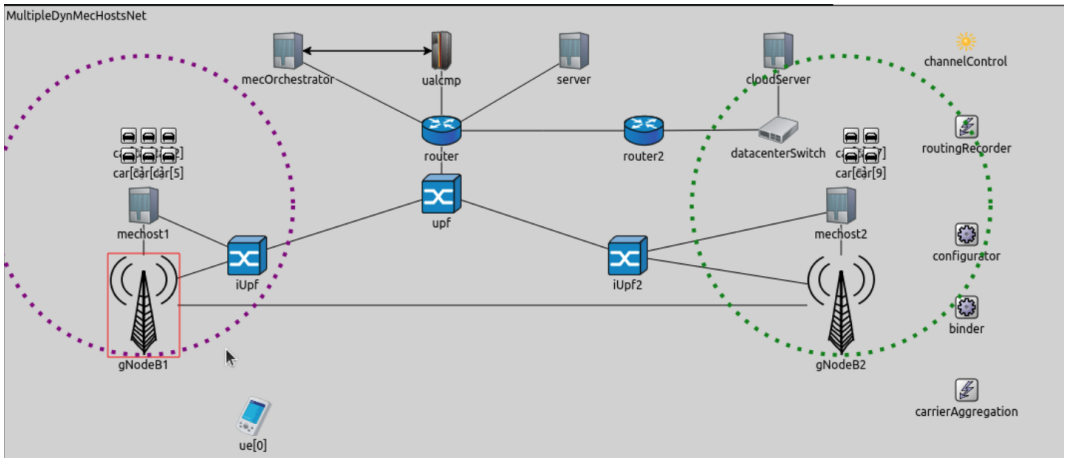


Figure 4.9: OMNeT++ Simulation Environment

hem[2]. Finally, this dataset is used in an experimental analysis to assess the model's performance under dynamic conditions, i.e., vehicles leaving the parking lot when running MEC Apps.

Figure 4.9 illustrates an example of the vehicular computing environment, which might be created using this simulation model in OMNeT++. The experiments were conducted using a 5G standalone network environment with a numerology index of $\mu = 2$. The network setup includes a single MEC-H connected to a gNB, simulating a scenario in a parking area near the gNB. A basic reward scheme was implemented for resource acquisition, where participating vehicles received integer values to represent the reward for joining. The MEC App deployed in each simulated scenario is the *MECWarningAlertApp* provided in the Simu5G library, designed to notify users when their vehicle enters a hazardous area (e.g., dense fog, icy roads, etc.). Users requesting the MEC App follow a linear mobility model from the INET library, moving at a constant speed of 10 m/s. The experiments were executed on a Linux Virtual Machine running OMNeT++ with 16 CPUs and 64 GB of RAM.

Model Validation

To validate the effectiveness of the proposed model, the network-induced delays across three service delivery schemes, namely at the cloud, edge of the network, and vehicle-based (far-edge), are evaluated. In the cloud scheme, the MEC Apps are hosted on a simulated cloud data center, which requires data transmission across the 5G RAN, the core network, and the edge. The second scheme assesses delays when MEC Apps run directly on the MEC-H, thus those due to 5G RAN and MEC local UPF co-located with the gNB (Figure 2.6). The last scheme involves host-

[2] https://parkeerdata.nl/opendata/arnhem/parkeergarages/transactiedata-parkeergarages

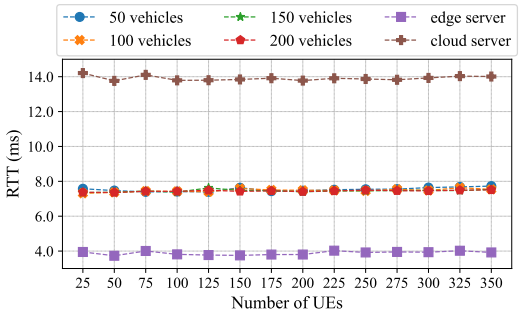
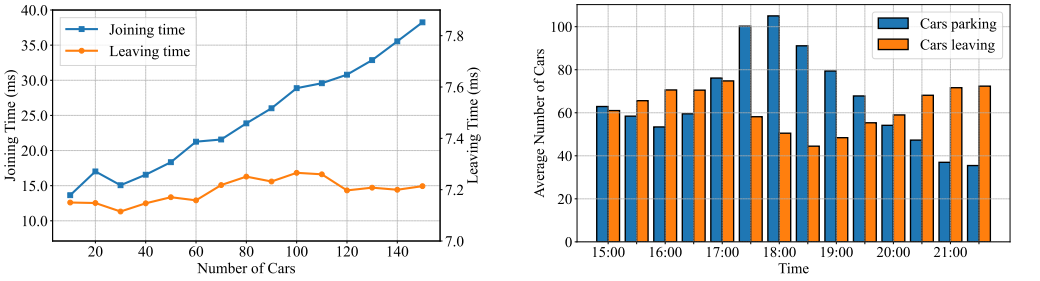


Figure 4.10: Round Trip Time variation with varying number of UE requests in the considered service delivery scenarios

ing MEC Apps on onboard vehicle resources, represented by the far-edge node in the extended architecture (see Figure 4.1). In this case, delays include data transmission between the network (gNBs) and devices (i.e., vehicle running MEC Apps and UE requesting their execution).

Figure 4.10 shows Round Trip Time (RTT) for each scheme. The x-axis represents the number of clients (UEs) requesting the execution of a MEC App to assess various load scenarios. In this experimental setup, each UE requests the execution of a single MEC App upon entering gNB's coverage area, with all UEs spawning simultaneously to mimic a flash-crowd scenario. To ensure accurate results, each experiment was repeated five times.

For the third scheme (vehicle-based), different quantities of vehicles participating in the MEC-H resource pool were tested, specifically 50, 100, 150, and 200 vehicles. In this scenario, the MEC App execution triggered by the UEs requests is executed onboard the vehicle. This allows the analysis of system scalability and how RTT correlates with the number of applications deployed on each vehicle. The results show that latency remains stable, even with more than 500 devices (i.e., 350 UEs and 200 vehicles) within the coverage of the MEC-H associated gNB, and is almost independent of the number of vehicles within the range of practical interest (note that computing-related delays are not considered in the reported simulations). Deploying MEC Apps on vehicles significantly reduces RTT by approximately 50% compared to cloud-based deployment. On the other hand, the far-edge scheme increases the RTT by $3ms$ compared to the edge mode, as it fully exploits wireless communications via the 5G network infrastructure. However, it should be noted that Simu5G handles V2V communications through the gNB base station, thus employing a network-mediated communication model also in V2V scenarios (5G Mode 1). Adopting 5G's sidelink mode (5G Mode 2), which allows direct V2V communication without gNB involvement, is expected to reduce RTT in future implementations.



(a) Resource acquisition and releasing protocol times

(b) Ceentral Garage City of Arnhem entering and leaving vehicles

Figure 4.11: Resources management times and vehicle distribution in a parking lot

Resource Management

As described in section 4.1.3, resource management involves the operations for acquiring, allocating, and releasing remote resources.

Figure 4.11a illustrates the time required by the protocols to collect and release resources from vehicles as they enter/leave the parking lot within the MEC-H AoI. The joining time represents the interval during which the MEC-H detects the availability of a new vehicle for MEC App allocation (step (1)-(6) in Figure 4.2a). On the other hand, the release time measures the period needed for the MEC-H to remove the vehicle from the resource pool (steps (1)-(4) in Figure 4.2b). The data reveals that the joining time increases progressively from 13 to 40 ms as more vehicles participate in the resource acquisition process, while the release time remains relatively stable at around 7 ms. This difference in performance can be attributed to the distinct communication overheads between the two protocols. The resource release process is straightforward, requiring only a few messages to remove a vehicle from the pool. In contrast, resource acquisition involves multiple request/response messages, as the device-initiated reward scheme requires the vehicle to request and negotiate available rewards.

While the simultaneous arrival of many vehicles in a parking lot is unlikely under typical conditions, such a scenario may occur during special events like festivals or football matches. To support this claim, data from three parking garages in the city of Arnhem, available on the Open Parkeerdata portal, were analyzed. Figure 4.11b shows the average number of vehicles entering and leaving the most frequented garage during rush hours. The data reveals that the number of parked vehicles reaches nearly the maximum considered in the test setup between 17:30 and 18:30. However, it is important to note that the peak of vehicle participation does not occur simultaneously, as the data were sampled at 30-minute intervals.

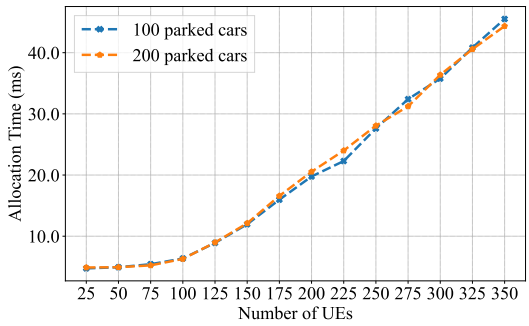


Figure 4.12: Resource Allocation Time

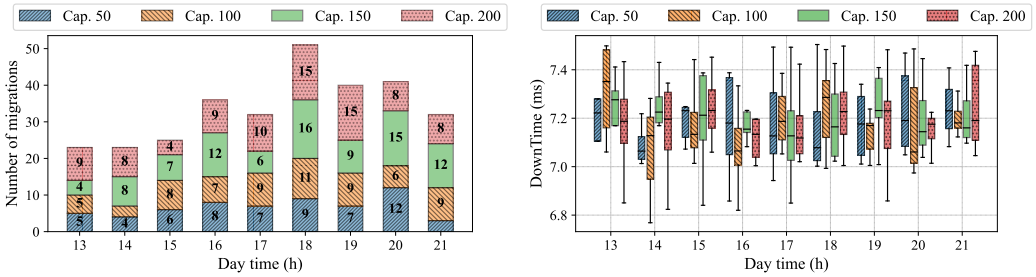
For resource allocation analysis, the time required to deploy MEC Apps on remote resources was measured based on the interactions between the VI and the VIM during steps (7)-(8) of the process shown in Figure 4.3. The simulation involves multiple UEs requesting MEC App execution and various numbers of parked cars belonging to the MEC-H resource pool. MEC Apps were distributed across remote nodes using a Round Robin scheduler. The simulation was repeated 10 times, varying the number of UEs and parked cars.

Figure 4.12 demonstrates that resource allocation delays follow an exponential growth pattern, influenced primarily by the number of MEC Apps deployed on parked vehicles. This is confirmed by the overlapping curves, which indicate that the delay values remain relatively constant even when the number of parked cars varies. It is worth mentioning that the simulation modeled a worst-case scenario in which all UEs requested MEC App execution simultaneously, thus leading to a substantial increase in network traffic. Despite this, the delay caused by these interactions remained negligible, with a delay of approximately 40 ms, even when the number of requests exceeded 300.

Migration Study

As previously mentioned, vehicles may leave the resource pool while their resources are still allocated, leading to potential service disruptions. The proposed architecture initiates a migration procedure to address this issue using the extended AMS described in section 4.1.4. In this context, the MEC-H transfers running MEC Apps from the departing vehicle to another available host. However, migrating stateful MEC Apps may result in a downtime period, during which the application becomes temporarily unavailable.

A realistic scenario involving vehicle volatility and UE activities within a parking area was developed to evaluate the performance of this migration process. The scenario requires data on vehicle activity (i.e., vehicles entering and leaving the AoI) and UE usage of network services in the



(a) Migration frequency

(b) Downtime

Figure 4.13: Migration related metrics

surrounding area. However, no datasets directly capture the required information for this specific context. Therefore, two real-world datasets were used: the first is the Arnhem parking garage dataset employed in the resource management analysis, and the second is the Bologna public WiFi network dataset[3], which records the number of users connecting to the WiFi network each hour.

To simulate the desired dynamics, the two datasets were analyzed to identify correlations between parking garage activity and network usage. As for the previous analysis, the “Central Garage" was chosen due to its capacity to host over 1,000 vehicles and its proximity to Arnhem’s train station. Data from Bologna’s WiFi networks near the city’s train station were then filtered to match this context. After pre-processing the data, key metrics were extracted, including vehicle occupancy time (i.e., the duration a vehicle remains parked), the average number of vehicles entering the garage per hour, and the average number of UEs connecting to the network per hour. Occupancy time was modeled using a normal distribution with a mean $\mu = 202.80$ minutes and a standard deviation $\sigma = 135.07$. The number of vehicles entering the garage and UEs connecting to the network was modeled using a Poisson distribution, with the Poisson interval set to 3,600 seconds. The average rate (λ) was adjusted based on the hour of the day.

The obtained distributions have been included in the simulation model to generate realistic vehicle entries and UE requests for MEC App execution. The stateful MEC used in the simulation, provided by the Simu5G library, defines a circular geographic warning zone and notifies the user whenever their vehicle enters or exits this zone.

Figure 4.13a illustrates the number of migrations triggered by adopting the described distributions. Four parking lot capacities (50, 100, 150, and 200 vehicles) were considered to scale the Poisson-distributed data from the original dataset. As MEC Apps were distributed across parked vehicles using a Round Robin algorithm, lower parking capacities led to a higher number of mi-

[3] https://opendata.comune.bologna.it/explore/dataset/iperbole-wifi-affluenza/information/

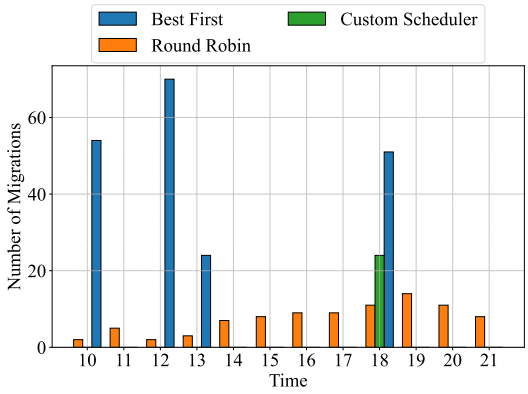


Figure 4.14: Scheduling Algorithms Comparison

grations, particularly during peak hours at 15:00 and 20:00. It is important to note that the number of migrations is also influenced by user activity and the number of vehicles present at different times of the day. These factors, generated through probability distributions, result in varying migration events depending on the number of UE requests and available vehicle resources, which are not detailed in this paper. The analysis focused on the interval from 13:00 to 21:00, corresponding to peak network activity hours.

The downtime has been measured as the time interval between the shutdown of the app on the leaving host and the end-user receiving the new MEC App location (refer to section 4.1.4 for each step details). Generally speaking, the downtime could be affected by the latency between the two involved entities, i.e., remote and local VI, and bandwidth [133]. In the experiments, latency was primarily affected by the distance between the MEC-H and the gNB, as well as 5G radio delays. Bandwidth did not significantly impact service interruption times, as the state data transmitted by the MEC Apps was smaller than 30 bytes.

As shown in Figure 4.13b, the downtime remains stable at around 7 ms. Overall, despite the number of MEC App relocations (Figure 4.13a) and the network activity increase, the downtime remains stable when migrating MEC Apps from remote host to local MEC-H resources.

4.2.4 Simulation Model Use Case: Minimize Task Migration

A custom scheduling algorithm was developed and tested using the simulation tool presented in this section. This algorithm aims to minimize the number of application migrations caused by vehicles leaving the parking lot while running MEC Apps.

To simulate vehicle and user behaviors, the scenario from the *Migration Study* in the previous section was recreated. This involved generating a series of Poisson and Gaussian distributions based on the two real-world datasets introduced earlier.

The simulation replicated vehicle and user activity over a 24-hour period, with a parking lot capacity of 150 vehicles. Three scheduling algorithms were tested: *Best First*, *Round Robin*, and the *custom algorithm*. The performance of each algorithm was evaluated based on the number of migrations it generated, as a lower number of migrations indicates improved reliability of MEC Apps. For clarity, the results were focused on periods of highest activity.

Figure 4.14 reports the associated performance results by referring to the most challenging case of the day hours with the highest user and vehicle activity levels. The *Best First* algorithm selects the first available vehicle in the pool with sufficient resources to run the application. This approach often leads to many migrations, as the selected vehicles may leave the parking lot while still executing applications. For instance, at the 12th hour of the simulation, the number of migrations exceeded 60. In contrast, the *Round Robin* algorithm distributes applications evenly across vehicles in the MEC-H resource pool, resulting in a more stable number of migrations, averaging around 7.42 throughout the simulation.

A custom scheduler was developed using the simulation platform, leveraging multiple Gaussian distributions derived from the Arnhem dataset after a pre-processing phase. This generated average vehicle occupancy times based on 10-minute interval samples. For each vehicle in the resource pool, the custom scheduler predicts its remaining time using these distributions and the time the vehicle joined. MEC Apps are assigned to vehicles with the longest predicted residency times.

As shown in the figure, the results indicate that the custom algorithm significantly outperforms the others in this scenario. While the algorithm could be further optimized using more advanced machine learning techniques, even in its current form, it reduces the number of migrations to around 20 by the 18th hour of the simulation. This demonstrates its effectiveness in improving application reliability by minimizing unnecessary migrations.

5 Enabling Vehicular Computing through MEC and O-RAN integration

As mentioned in section 2.5, the O-RAN Alliance defined a series of standards to make the RAN open. It extends the 3GPP standardized base stations to make them able to support two distinct RICs, programmable platforms designed to execute optimization algorithms and enable closed-loop control of the RAN [48, 54].

This Chapter examines how the synergies between ETSI MEC and O-RAN can be exploited to create a highly programmable, edge-computing-enabled ecosystem, all while maintaining compliance with both telecommunications standards. The discussion emphasizes how this convergence can significantly enhance the practical implementation of the vehicular computing paradigm introduced in Chapter 4.

Furthermore, this Chapter places particular emphasis on the edge layer, highlighting the central role of xApps in this integration. To address the current challenges in the xApp development process, the Chapter presents a detailed design and implementation framework that reduces the complexity of xApp development, streamlines the process, and encourages innovation in the programmable RAN environment.

5.1 Leveraging O-RAN networks to support MEC implementations

The integration of ETSI MEC with 5G networks has been widely researched over time, yielding significant benefits for both the academic community and the advancement of emerging technologies [47, 134, 135]. This synergy drives innovation across key sectors such as autonomous vehicles, innovative city applications, healthcare solutions, and the augmented/virtual reality market while contributing to broader industry growth [134].

Both ETSI and 3GPP entities play a central role in enabling the integration of MEC within the 5G infrastructure, particularly in areas such as traffic routing, local breakout, and policy control operations. As described in section 2.3, these alliances define enablers that facilitate the interaction between the two technologies. Xavier *et al.* [136] presented an integration of MEC and the 5G core through their proposed communication API. They developed a MEC App that exploits these APIs to demonstrate their approach's feasibility, highlighting how the MEC system can interact dynamically with the 5G core network to optimize application performance and user experience.

Similarly, Tomaszewski *et al.* [135] explored the challenges of managing network slices in a MEC-compliant 5G environment, a critical aspect for ensuring that different applications receive appropriate resources in line with their service-level agreements. The dynamic allocation and management of network slices is crucial, especially for latency-sensitive applications such as V2X communications. Wadatkar *et al.* [137] emphasized how V2X could benefit from integrating edge computing and 5G, showcasing how real-time data processing at the edge can improve vehicular communications by reducing latency and enhancing safety-critical functions.

While integrating MEC with 5G has tremendous potential to transform industries and enhance next-generation applications, it also poses significant challenges in terms of resource management, security, heterogeneous traffic coexistence, and advanced technology integration. Huang *et al.* [138] made early contributions by developing a framework for traffic redirection in MEC within 5G environments. Still, their work lacked attention to the fine details of standardization, especially about ETSI compliance.

A promising approach to addressing these challenges involves leveraging O-RAN standards. O-RAN offers a more flexible and programmable framework, which can be exploited for integrating MEC with 5G, especially by enabling greater control over the RAN. The authors in [139] showed a first draft of an O-RAN-centric integration of ETSI MEC in 5G networks. However, their work was limited by the lack of full O-RAN specifications availability at the time and thus only presented a preliminary indication of what such an integration could achieve. The following sections of this dissertation provide a design overview of how these two standards can be integrated, highlighting their mutual benefits and the challenges this integration presents.

5.1.1 Design Overview

Figure 5.1 illustrates the integration of the ETSI and O-RAN standards presented in this dissertation. Starting at the MEC system layer, the O-RAN SMO framework manages orchestration and monitoring of the RAN controller, hosting the Non-RT RIC and providing interfaces to support interaction with various network components. On the ETSI MEC side, the MEC-O acts as a resource orchestrator, maintaining a global view of the MEC system, including deployed

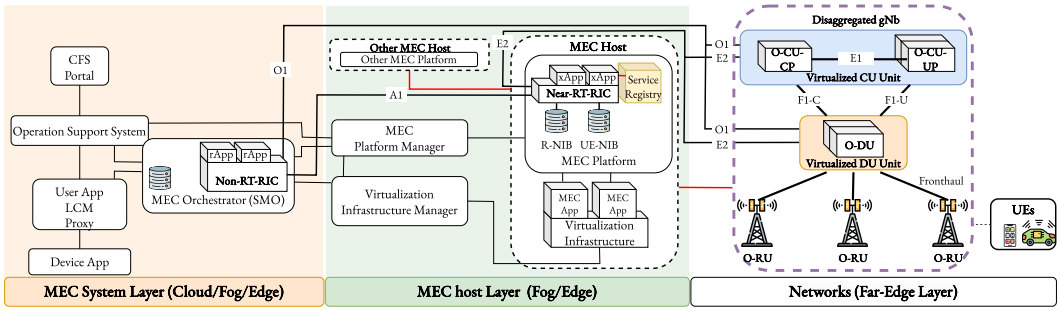


Figure 5.1: Integration ETSI MEC and O-RAN standards

MEC-Hs, available resources, services, and network topology. It selects the appropriate MEC-H for application instantiation based on specific constraints. Both the SMO and MEC-O operate at the same layer, having a comprehensive view of the network infrastructure and managing their respective components. Therefore, they may collaborate to ensure cooperation between MEC and O-RAN orchestration methods.

At the lower level, the Near-RT RIC enables xApp execution in near real-time. xApps are modular components that implement custom logic, receiving telemetry from the RAN and sending control commands via the E2 interface. This supports closed-loop control of the RAN at a near real-time scale (below 1s). On the ETSI MEC side, MEC Apps discover and consume MEC services to meet user requests. As described in section 2.3, MEC Apps can leverage several ETSI MEC services, such as the RNI service [140], which provides access to radio conditions and user plane measurements, and the V2X information service [141], which delivers relevant V2X data from the 3GPP network.

xApps can seamlessly supply the data provided by these services. The proximity of xApps allows to offer MEC-compliant services to MEC Apps through the service registry. Additionally, the Near-RT RIC, which is built on a microservices architecture (refer to section 2.5), shares a similar approach with VI platforms that provide compute, storage, and network resources to MEC Apps. As the Near-RT RIC runs on Kubernetes, it can also host MEC-compliant applications, service registries, and MEC services alongside xApps.

This dissertation focuses on the edge layer of the architecture, specifically on components included in the MEC Host Layer. This integration enables the execution of user-requested applications, MEC Apps, that do not modify RAN configurations. Instead, MEC-compliant applications run alongside xApps and rApps on the RICs, leveraging their functionality to deliver services to end-users. Additionally, this integration creates opportunities for RICs managed by different mobile network operators to interact and may enable xApps to connect with the network core [136].

5.1.2 Challenges

The integration of ETSI MEC and O-RAN standards presents significant opportunities for improving edge computing and RAN management. However, key challenges need to be addressed to fully unlock the potential of this integration. Two of the primary challenges are as follows:

1. **Unifying the interfaces to enhance the capability of both standards**: A major challenge in integrating ETSI and O-RAN standards lies in the need to unify their interfaces. This unification aims to eliminate ambiguities in their interaction, interworking, and integration processes. Without clear, unified interfaces, these standards risk operational inefficiencies and miscommunication between network components, which could hinder the deployment of edge services and RAN control mechanisms. A first approach to this topic has been explored in [142], offering a solution that focuses on the SMO framework that ensures the unification of interfaces O-RAN, 3GPP and ETSI specifications.

2. **Immaturity in the existing framework for xApp development**: Another significant challenge arises from the underdeveloped framework for xApp development, mostly due to the complexity of the E2 interface. The E2 interface, which facilitates communication between the RIC and RAN, offers great flexibility through its multiple Service Models (SMs). However, this flexibility introduces complexity, making it difficult for developers to create xApps efficiently. Existing frameworks do not entirely abstract the intricacies of E2SMs, leading to longer development times and potential errors in implementation.

This dissertation focuses on this second challenge by proposing and prototyping a new framework that simplifies xApp development. The framework, presented in the next section, allows developers to concentrate on the core control logic of their xApps while abstracting the complexities of the E2 interface and its service models behind a simple API. Doing so reduces the barriers to xApp development, accelerating innovation and deployment within the O-RAN ecosystem.

5.2 xApp Framework for O-RAN E2 Service Models

As discussed in the previous section, the xApp development process remains in its early stages, primarily due to the complexity of the E2 interface. This complexity limits developers from fully focusing on xApp logic, as they also need to deal with the intricacies of this interface. This section presents the design and implementation details of xDevSM, a framework specifically designed to simplify xApp development for the OSC Near-RT RIC.

xDevSM offers a set of APIs that abstract the steps defined in various E2SM protocols, streamlining the interaction between the xApp, the Near-RT RIC, and the E2 termination at the RAN

node. By handling the complexities of configuring E2SM messages, xDevSM allows developers to focus entirely on defining their application logic.

The framework builds on OSC RIC components and extends xApp functionality with a wrapper that can interface with shared libraries implementing specific E2SM protocols, such as KPM. As this section demonstrates, xDevSM has been thoroughly tested with various open-source RAN implementations, including srsRAN, OAI, and NVIDIA's Aerial RAN CoLab (ARC), demonstrating both its flexibility and its ability to validate functional use cases.

5.2.1 Related Work

Previous research has demonstrated that dynamically configuring the RAN stack through closed-loop control using xApps can result in significant improvements in spectrum utilization, throughput, and user satisfaction [143, 144, 145, 146]. The role of RICs has been explored in several areas, including network slicing [143], load balancing and handover [147, 148], traffic shaping [146], and spectrum sharing [149], among others [54].

Despite the advancements made through multiple E2 and RIC projects, developing xApps that work across different components developed by various projects or vendors remains a challenge. Additionally, the rapid cycle with which the O-RAN ALLIANCE publishes new specifications often conflicts with the time required to develop new OSC releases, complicating the maintenance and update of the E2 specifications. To address this, some efforts have implemented abstraction layers over the E2 interface [150], while others have simplified SMs by using Protobuf buffers instead of ASN.1 data structures [151, 152]. Although this approach simplifies the development and testing of new SMs, it lacks full O-RAN compliance and requires both the xApp and RAN to support Protobuf. Additionally, Protobuf encoding introduces a higher overhead compared to the binary encoding offered by ASN.1 [153].

To address the abovementioned challenges, this dissertation proposes a flexible framework that abstracts away the complex E2SM encoding/decoding processes, facilitating seamless interaction with multiple RAN implementations.

5.2.2 xDevSM: A Flexible SM Framework for xApp Development

As section 2.5 mentions, the OSC offers a comprehensive set of libraries and functions for various programming languages designed for their Near-RT platform version to support xApp development. This suite includes APIs for messaging through the RMR, such as callback for registration, E2AP encoding and decoding, subscription APIs, and the Shared Data Layer (SDL) interface. xDevSM framework leverages the Python-based framework [154] provided by OSC, which in-

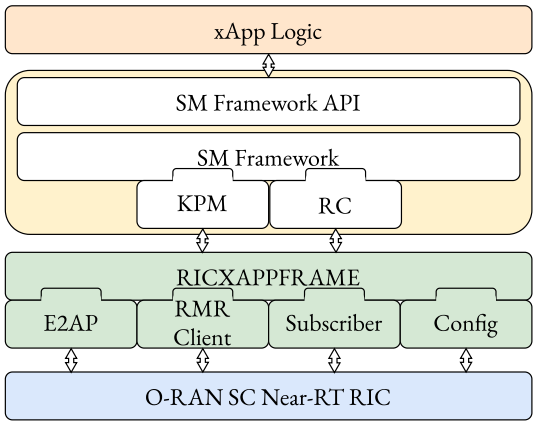


Figure 5.2: xDevSM and OSC RIC components. The xDevSM components are in yellow, while the OSC frameworks are in green and blue. The xApp developer only takes care of the xApp logic, in orange

troduces two main types of xApps: Reactive xApps (`RMRXApp`) and general xApps (`XApp`). Reactive xApps respond to incoming RMR messages through registered callbacks, allowing the main xApp to manage multiple message types. General xApps, on the other hand, require the main xApp to actively fetch messages from the RMR buffer.

While the OSC frameworks are primarily focused on facilitating communication between RIC components, they do not provide APIs that simplify E2SM-related interactions with the E2 node. Once the xApp subscribes or connects to an E2 node, developers must handle the complex task of serializing and deserializing E2SM messages, making them responsible for understanding the specific E2SM versions supported by the E2 nodes to build functional xApps.

xDevSM addresses these challenges by providing a plug-and-play solution for encoding and decoding E2SM data through simple, well-defined APIs. As illustrated in Figure 5.2, the framework is built on top of the Python xApp framework provided by the OSC [154, 155] and comprises two components: the SM Framework and the SM Framework API.

The SM Framework offers Python objects and functions that enable accurate decoding and encoding of messages related to supported E2SMs. These functions are implemented in shared libraries written in C, which are dynamically loaded by the framework as needed. In this initial version, the shared libraries are based on E2SM functions from the FlexRIC project [56], along with custom functions to streamline common xApp tasks requiring data encoding/decoding. Python's garbage collector is leveraged to manage memory deallocation for the corresponding C structures. Developers can extend the shared libraries with additional encoding/decoding functions to support new E2SMs and then integrate them with Python objects. The API component provides interfaces for interacting with the Near-RT RIC and E2 nodes, allowing developers to imple-

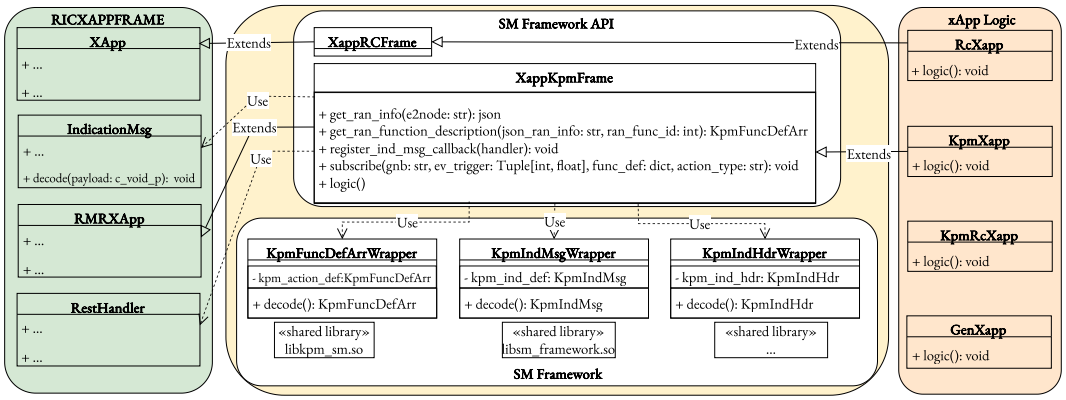


Figure 5.3: Class diagram for xDevSM

ment their xApp logic. It includes an abstract method that xApp developers must define, called after xApp initialization, and contains the application-specific logic. For example, it can request the decoded RAN function description or store the key performance indicators in a time-series database. Additionally, the xApp containing the logic can receive decoded KPM messages and can directly send RC messages in an encoded form. These messages are managed using the SM framework functionalities described above and sent to the RIC components using the OSC framework.

Compared to other solutions [156], xDevSM relies on shared libraries to manage ASN.1 serialization and deserialization procedures, leveraging the C language, which is the most widely adopted in this context [56, 157, 158]. This approach enhances the robustness of xDevSM, as it simplifies the process of finding and building shared libraries to support new service models. Additionally, the extensive use of C in this environment ensures better performance and compatibility, making xDevSM a more reliable and adaptable framework for xApp development.

5.2.3 xDevSM: Design and Implementation

The new abstraction layer introduced in the xDevSM framework facilitates data translation and optimizes memory management by dynamically handling the allocation and deallocation of external structures. This dissertation provides details on the KPM SM [159] encoding/decoding features provided in this layer by xDevSM, as the RC SM is currently under development. As shown in Figure 5.3, the API component includes a class called `XappKpmFrame`, which extends the `RMRXApp` class provided by the OSC xApp framework [155]. This extension adds methods for retrieving E2 node information, such as the connection status to the Near-RT RIC and available RAN functions, as well as subscription-related methods. To facilitate RAN information retrieval, the framework utilizes a Python object called `KpmFuncDefArrWrapper`, which refers to a custom data structure defined in the `libsm_framework.so` shared library. This structure is essential for decoding

Listing 5.1: Logic of a Basic KPM xApp

```
class KpmXapp(kpmframe.XappKpmFrame):
    def logic(self):
        self.register_ind_msg_callback(handler=self.indication_callback)
        # Registring other callbacks and get list of available gnb
        for index, gnb in enumerate(gnb_list):
            json_obj = self.get_ran_info(e2node=gnb)
            if json_obj["connectionStatus"] != "CONNECTED":
                continue

            ran_function_description = self.get_ran_function_description(json_ran_info=json_obj)
            func_def_dict = ran_function_description.get_dict_of_values()

            func_def_sub_dict = {}
            selected_format = format_action_def_e.END_ACTION_DEFINITION
            if len(func_def_dict[format_action_def_e.FORMAT_4_ACTION_DEFINITION]) == 0:
                selected_format = format_action_def_e.FORMAT_1_ACTION_DEFINITION
            else:
                selected_format = format_action_def_e.FORMAT_4_ACTION_DEFINITION

            if selected_format == format_action_def_e.END_ACTION_DEFINITION:
                self.terminating_xapp()
                return

            # Selecting only supported action definition
            func_def_sub_dict[selected_format] = [value for value in func_def_dict[selected_format] if value in
                  measurements_ids]

            # Sending subscription
            ev_trigger_tuple = (0, 1000)
            status = self.subscribe(gnb=gnb, ev_trigger=ev_trigger_tuple, func_def=func_def_sub_dict)

            # Check Status...
```

and building RAN function definitions, which are stored in hexadecimal and E2SM-encoded formats by the Near-RT RIC subscription manager. The wrapper also handles memory deallocation once the Python object is no longer needed.

The `subscribe` method of the `XappKpmFrame` class allows developers to specify the E2 node, the RAN functionalities to be monitored, and the reporting period for the xApp subscription. Before sending this information to the subscription manager, the `subscribe` method encodes the RAN functionalities and the reporting period using functions defined in the `libsm_framework.so` shared library. Upon receiving this encoded information, the subscription manager performs additional encoding using the E2AP libraries and then forwards the subscription to the E2 nodes. The encoding functions in `libsm_framework.so` rely on procedures defined in `libkpm_sm.so`, which is built using the FlexRIC [56] framework.

Developers can exploit this framework by either extending the `XappKpmFrame` class or by encapsulating an instance of it. Extending the `XappKpmFrame` offers greater flexibility, allowing developers to customize behaviors of internal methods such as post-initialization actions, logger level adjustment, and modified decoding functions. On the other hand, encapsulating an instance allows

Listing 5.2: gNB Registration on Near-RT RIC

```
[
    {
        "inventoryName": "gnb_001_001_00000e05",
        "globalNbId": {
        "plmnId": "00F110",
        "nbId": "000000000000000000000111000000101"
        },
        "connectionStatus": "CONNECTED"
    }
]
```

developers to utilize the class's procedures without changing its internal behavior, simplifying integration and reducing potential errors.

This dissertation explores the first option, as illustrated in Figure 5.3 (orange block). The `KpmXapp` class inherits from `XappKpmFrame`, registers a callback method to handle *Indication Messages*, and defines the `logic` method. This ensures that all operations related to the dispatching of indication messages and decoding the information are handled within the framework. Specifically, the framework automatically decodes the E2AP part using the `RICXAPPFRAME IndicationMsg` class (green block Figure 5.3) and the E2SM data using the `KpmIndMsgWrapper` and `KpmIndHdrWrapper` classes (yellow block Figure 5.3). The E2SM-related classes expose a `decode` API, which invokes a function defined in the `libkpm_sm.so`. This function decodes the E2SM-related information and returns a Python Object corresponding to the KPM related data. This modular design allows for easy updates to the E2SM by simply changing the shared library version, provided that function signatures remain consistent across different E2SM versions. Once the received message has been decoded, the registered function containing the behavior defined by the xApp developer is executed.

The `logic` method is an abstract method provided by the `XappKpmFrame` that allows the developers to define the xApp behavior. Within this method, developers can specify which E2 node to select based on a particular logic, identify RAN functionalities of interest, set reporting period, and more. For instance, in the scenario provided in Listing 5.1, after registering a callback and retrieving the available gNBs, the xApp fetches KPM parameters associated with each gNB, selects formats supported by framework, and sends a subscription to the gNB. The xApp then receives notifications for each indication event in its decoded form. This example highlights how the xDevSM framework simplifies the process for developers when designing and implementing an xApp using existing frameworks.

Listing 5.3: gNB RAN Function Descriptions Decoded in xApp

```
{[...], "id": "ricxappframe.xapp_frame", "mdc": {}, "msg": "Available functions: {0: [], 1: [], 2: [], 3: ['DRB.
    PdcpSduVolumeDL', 'DRB.PdcpSduVolumeUL', 'DRB.RlcSduDelayDl', 'DRB.UEThpDl', 'DRB.UEThpUl', 'RRU.PrbTotDl',
    'RRU.PrbTotUl'], 4: []}"}
{[...], "id": "ricxappframe.xapp_frame", "mdc": {}, "msg": "Selected functions: {3:['DRB.PdcpSduVolumeDL', 'DRB.
    PdcpSduVolumeUL', 'DRB.UEThpDl', 'DRB.UEThpUl', 'RRU.PrbTotDl','RRU.PrbTotUl']}"}
```

Listing 5.4: gNB KPM parameters Decoded by xDevSM

```
{"ts": 1738666526341, "crit": "DEBUG", "id": "ricxappframe.xapp_frame", "mdc": {}, "msg": "printing info ue[0]"}
{"ts": 1738666526341, "crit": "INFO", "id": "ricxappframe.xapp_frame", "mdc": {}, "msg": "DRB.PdcpSduVolumeDL:0"}
{"ts": 1738666526341, "crit": "INFO", "id": "ricxappframe.xapp_frame", "mdc": {}, "msg": "DRB.PdcpSduVolumeUL:0"}
{"ts": 1738666526341, "crit": "INFO", "id": "ricxappframe.xapp_frame", "mdc": {}, "msg": "DRB.RlcSduDelayDl:0.0"}
{"ts": 1738666526341, "crit": "INFO", "id": "ricxappframe.xapp_frame", "mdc": {}, "msg": "DRB.UEThpDl:0.024"}
{"ts": 1738666526341, "crit": "INFO", "id": "ricxappframe.xapp_frame", "mdc": {}, "msg": "DRB.UEThpUl:0.552"}
{"ts": 1738666526341, "crit": "INFO", "id": "ricxappframe.xapp_frame", "mdc": {}, "msg": "RRU.PrbTotDl:74"}
{"ts": 1738666526341, "crit": "INFO", "id": "ricxappframe.xapp_frame", "mdc": {}, "msg": "RRU.PrbTotUl:166"}
```

5.2.4 Analyzing xDevSM with different Open-Source RANs

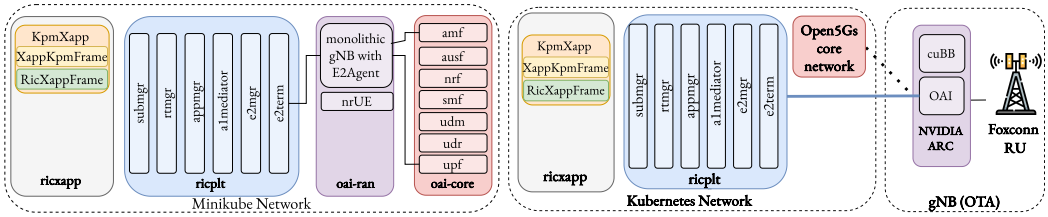
This section explores the flexibility of the xDevSM framework across various RAN implementations. The evaluation includes experiments conducted with both OAI and srsRAN, tested in two distinct scenarios: simulated environments and OTA.

OpenAirInterface

This section presents an analysis based on deployments utilizing OAI software, including both simulated environments and real-world scenarios where the OAI RAN operates alongside the Open5GS core and Commercial Off-the-Shelf (COTS) user equipment (UEs).

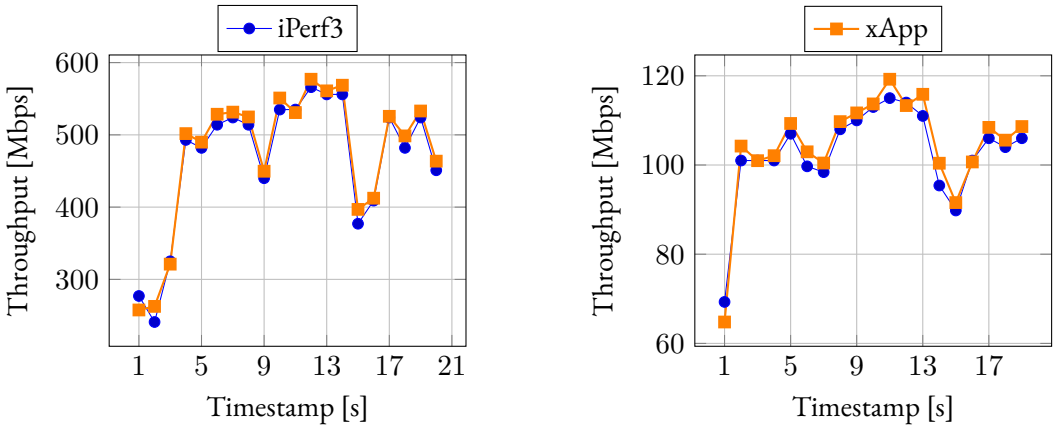
Simulated Deployment. As illustrated in Figure 5.4a, the deployment involves setting up a 5G network using Minikube, where OAI provides both the RAN and core [1].To enable the E2 interface and establish connectivity with the near-RT RIC, the OAI RAN was configured with its default E2 agent, provided by the FlexRIC project. A pod within the `oai-ran` namespace implements an NR UE that connects to the base station via OAI's RFSim. This setup creates an O-RAN-compliant, isolated environment, allowing for efficient testing and prototyping of xApps. The KpmXapp (Listing 5.1) was employed to validate xDevSM compatibility with OAI. In this configuration, the SM framework utilizes the shared library (`libkpm_sm.so`) produced during the building phase of the E2Agent from the FlexRIC SDK. Multiple tests confirmed effective communication between OAI software and OSC Near-RT RIC Releases I and J, while the xApps performed their designated tasks as expected.

[1] `https://github.com/aferaudo/ORANInABox/wiki`



(a) OpenAirInterface deployment in Minikube (b) OpenAirInterface over the air

Figure 5.4: OpenAirInterface Deployments



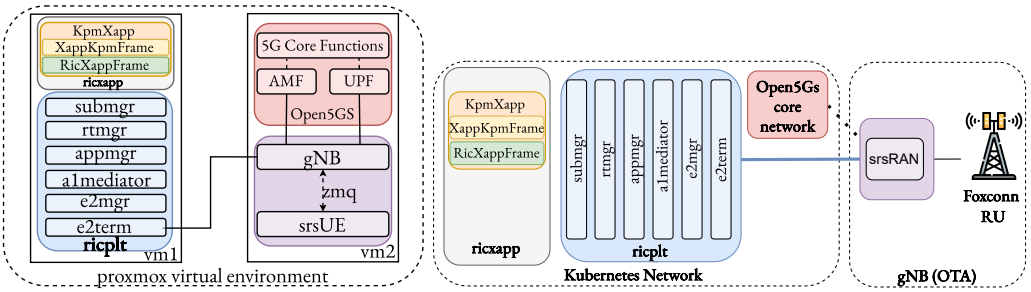
(a) Downlink (b) Uplink

Figure 5.5: Comparison of iPerf3 (blue dots) and xApp (orange squares) reported values with OAI

Listing 5.2 shows the successful connection of a base station to the Near-RT RIC. Listing 5.3 presents an example of the *RAN Function definitions* received from the E2 node, processed by the xApp, and decoded by xDevSM. Moreover, Listing 5.4 provides an example of information sent through the E2 interface by the gNB. These include the amount of Packet Data Convergence Protocol (PDCP) traffic exchanged during the interval defined in the KPM subscription (`DRB.PdcpSduVolumeDL`, `DRB.PdcpSduVolumeUL`), the average throughput (`DRB.UEThpDl`, `DRB.UEThpUl`), and the number of Physical Resource Blocks (PRB) allocated (`RRU.PrbTotDl`, `RRU.PrbTotUl`).

This simulated deployment provided the foundational environment for developing, testing, and prototyping the xDevSM framework. The shared libraries created in this environment were reused for subsequent deployments.

Over-the-Air Deployment with NVIDIA ARC. Building upon the simulated environment, this deployment involved an OTA setup. It leverages the X5G testbed [152], where OAI was employed as the CU and DU-high, while the DU-low (PHY-high) used NVIDIA ARC's inline acceleration via GPU. This DU is connected to a Foxconn RU operating in the 3.7-3.8 GHz band,



(a) srsRAN simulated deployment (b) srsRAN Over-The-Air

Figure 5.6: srsRAN Deployments

with a bandwidth of 100 MHz, and connected to Samsung S23 and OnePlus AC2003 Nord COTS smartphones. The default E2 agent for the OAI stack was again used in this setup, with Open5GS as the core network to validate functionality with different AMF and UPF implementations, which may encode UE information differently. Figure 5.4b illustrates the final deployment.

Tests were conducted with up to five connected UEs, and the xApp accurately reported metrics for all devices. Disconnection scenarios were also evaluated, with the xApp successfully removing disconnected UEs from the list of available devices sent from the gNB to the near-RT RIC.

Figure 5.5 shows that the throughput values reported by the xApp closely match the traffic generated by the UEs using iPerf3 over TCP, both downlink and uplink. A minor offset of approximately 2% was observed, attributed to a misalignment in reporting of the KPMs by the xApp and iPerf3 logs. This discrepancy also relates to the overhead introduced by headers in the TCP/IP stack and at PDCP, which is captured by the xApp and not by iPerf3, as iPerf3 measures application layer throughput, while the xApp tracks throughput within the RAN.

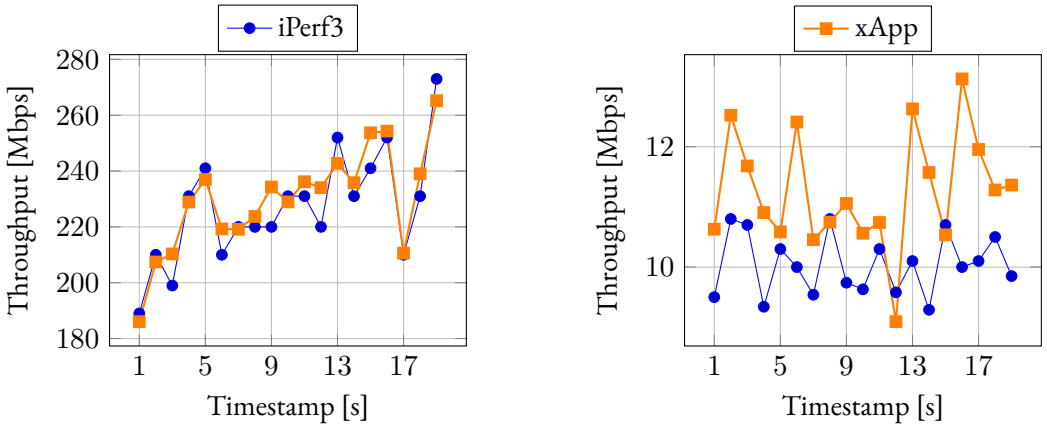
srsRAN

This section examines deployments using srsRAN to evaluate the effectiveness of xDevSM framework with a different RAN implementation. As with the previous analysis, the experiments are conducted in two phases: a simulated deployment using two virtual machines for the Near-RT RIC and 5G components, followed by an OTA deployment.

Simulated Deployment. In this deployment, the simulated setup from the srsRAN tutorial[2] is used, where the srsUE and gNB communicate via a ZeroMQ-based RF driver, with Open5GS as the core network.

As illustrated in Figure 5.6a, the setup differs from the one outlined in the tutorial. It relies on two virtual machines connected over the same network: one hosts all the 5G components (RAN,

[2] https://docs.srsran.com/projects/project/en/latest/tutorials/source/near-rt-ric/source/index.html



(a) Downlink

(b) Uplink

Figure 5.7: Comparison of iPerf3 (blue dots) and xApp (orange square) reported values with srsRAN

UE, and core), and the other runs the Near-RT RIC Release J in a single-node Kubernetes cluster. In this deployment, the E2Term SCTP service of the Near-RT RIC has been exposed to interact with srsRAN's E2 node. For testing, the shared library for the KPM v3.00 service model was built using the FlexRIC SDK, as srsRAN supports KPM v3.00 [159]. The xApp was then deployed using the xDevSM framework, utilizing the service model designed for this environment. It should be noted that, by default, srsRAN exposes more KPMs than OpenAirInterface.

Over-the-Air Deployment with 7.2 Split. The OTA deployment extends the simulated setup, utilizing srsRAN with the same Foxconn RU as shown in Figure 5.6b. The transmission occurs over a 40 MHz bandwidth, and iPerf3 tests are conducted similarly to the previous OTA scenario. As expected, the xApp decodes and displays metrics from the gNB correctly. Figure 5.7 compares the traffic generated by iPerf3 (using TCP) with the traffic reported by the xApp. The results show a 1% difference in downlink throughput and around 10% difference in uplink throughput. This last difference is because the throughput in the uplink is limited to 10 Mbps, and the xApp measures the performance at the PDCP level. Therefore, the overhead due to the TCP/IP and PDCP headers is higher than that of other experiments. Sometimes, the xApp failed to decode UEs updates when devices disconnected from the srsRAN base station. This issue was not observed with the OAI RAN implementation, suggesting that further investigation on the srsRAN side is necessary.

Listing 5.5: OAI gNB Exposed RC RAN Function Definitions

```
{"ts": 1739263269274, "crit": "INFO", "id": "ricxappframe.xapp_frame", "mdc": {}, "msg": "INVENTORY NAME:
      gnb_001_001_00000e00"}
[Event Trigger]
UE Information Change
[Report]
UE Information
[Control]
Radio Bearer Control
```

5.2.5 Challenges and Extensions in xDevSM: A Discussion

One of the key challenges in enhancing xDevSM[3] lies in its adaptability to evolving E2SMs. As O-RAN standardization advances, service models may undergo frequent modifications, necessitating a flexible and efficient approach to integration.

Currently, adapting xDevSM to new or modified E2SMs requires manual intervention. This process involves designing new Python objects corresponding to the underlying SM structures defined in the shared libraries (currently based on C). As this mapping process is manual, when changes in the E2SM notation occur, developers must manually modify the C code that bridges the E2SM and custom structures before rebuilding the shared library to reflect these updates. Additionally, developers need to define the Python counterpart of these structures. This manual process introduces potential for error, increases maintenance time, and limits scalability when supporting multiple service models across different E2SM versions.

Automating the translation between E2SM structures and Python objects will be a key area of improvement. This automation would allow xDevSM to dynamically adapt to changes in E2SM versions and reduce the need for extensive manual code adjustments, thereby increasing efficiency and reducing the likelihood of inconsistencies.

Another challenge involves the E2AP encoding and decoding, which xDevSM depends on from the OSC xApp framework. Discrepancies can arise between the xApp framework's E2AP implementation and the actual RAN E2AP implementations, leading to potential synchronization issues. To mitigate these synchronization risks, future development focuses on integrating continuous testing solutions for xDevSM, ensuring compatibility between the framework, the Near-RT RIC, and the RAN implementations.

Additionally, future research aims to extend xDevSM capabilities to include the RC service model. This SM enables the dynamic adaptation of parameters exposed by the RAN through the E2 interface (Listing 5.5), allowing near-real-time changes to RAN behavior. Such adaptability is

[3] https://github.com/wineslab/xDevSM

crucial to enable closed-loop control and advanced network management tasks, including RAN slicing, resource allocation optimization, and policy-driven reconfiguration.

For instance, Listing 5.5 provides an example of RC RAN function definitions decoded by xDevSM and exposed by a OAI gNB through the E2 interface. In this scenario, the gNB supports the *Radio Bearer Control* feature, which enables changes to the bearers of E2 nodes or specific UEs, such as those concerning QoS parameters. This capability is crucial for fine-tuning network performance and ensuring optimal service quality for different traffic profiles.

As xDevSM continues to evolve, addressing these challenges through automation and continuous integration will be key to expanding its applicability and ensuring its long-term viability in the evolving O-RAN ecosystem.

6 Conclusions and Future Work

The VC paradigm presents significant challenges due to the intermittent connectivity and high mobility of vehicles. This dissertation focused on addressing some of these challenges by tackling the increasing complexity of vehicular networks and edge computing within the VC ecosystem, intending to support next-generation vehicular applications effectively.

- In Chapter 3, this dissertation introduced two original solutions to improve communications reliability in VANET. The first one focused on mitigating VANET attacks through a robust reputation system that excludes vehicles exhibiting malicious behavior within the VANET. The dissertation provided comprehensive details on this solution's design, implementation, and integration within real-world 5G scenarios. Moreover, this was evaluated by building a dataset comprising standard V2V and V2I messages exchanged during an exceptional event, such as an accident. The second contribution explored the potential of leveraging in-vehicle TSN architecture for external communications. This resulted in a proposal for a TSN-like controller to reduce interference and prevent data loss in vehicular networks.

- In Chapter 4, this dissertation designed a novel VC architecture exploiting the ETSI MEC standard to lay the foundation of a cloud continuum of resources from the traditional (usually geographically distant) cloud to far-edge resources accessible in a standard way. The presented scheme allows devices/far-edge nodes to join the resource pool of edge nodes and device owners to access a reward system. It also addressed resource volatility problems as devices enter and leave the edge node resource pool. The feasibility of this architecture was also demonstrated through a simulation tool focusing on parked vehicles, validated by extensive experiments on response time, resource management, and migration. The dissertation also provided a case study of how this tool can be used to test resource allocation algorithms in minimizing task relocation.

- Chapter 5 outlined an in-depth investigation of integrating O-RAN with the ETSI MEC standard to facilitate MEC usage in 5G networks. This Chapter highlighted the synergies and the challenges identified for such an integration. Moreover, as xApps have a key role in

such an integration, the Chapter detailed the design and implementation of a framework, with the intent of minimizing development effort for O-RAN xApps. By introducing an abstraction layer to manage E2SM encoding/decoding, the framework allowed developers to focus on the control logic rather than service model intricacies.

In the next section, this dissertation outlines future work that builds upon the solutions presented, exploring further advancements and research directions to enhance vehicular networks and edge computing environments.

6.1 Envisioned and Planned Future Work

From this dissertation emerge many opportunities and future directions, some of which are identified in this section.

6.1.1 VANET Reliability

Building on the DIVA algorithm's ability to assign a reputation score to each vehicle transmitting road-related information (section 3.1.3), a future direction involves leveraging these scores as a criterion for reward allocation in VC environments. For example, in the architecture presented in Chapter 4, rewards could be dynamically defined in proportion to each vehicle's reputation. Additionally, exploring the integration of network-based reputation with computing-based reputation, which evaluates a vehicle's reliability in resource provisioning, could further refine reward mechanisms used in VC scenarios.

Additionally, since DIVA classifies messages according to the reputation score of each vehicle, it can serve as a labeling mechanism to identify and flag malicious behaviors within the vehicular network. Future research could apply this labeled dataset to train a machine-learning model, advancing the system's ability to autonomously detect other misbehaving vehicles. This approach would enable real-time misbehavior detection and model adaptability, allowing the machine-learning system to evolve in response to emerging malicious behaviors.

In the same direction, section 3.2 introduced a TSN-like Controller designed to improve QoS in vehicle platooning by orchestrating message dissemination over VANET. While the approach obtained promising results, further research is needed, particularly in managing dynamic events like vehicles exiting a platoon and reassigning communication slots. Addressing these gaps would enhance the system's resilience and communication efficiency in dynamic vehicular environments.

Moreover, future work could explore the integration of cellular and TSN domains to support time-sensitive applications and low-latency communication requirements, particularly in 5G infrastructures [95]. Thanks to the design of this solution, further studies can examine developing

intelligent strategies for handling platoon formation errors and supporting the prioritization of safety-critical messages. Implementing distinct priority-based queues could facilitate the reliable and rapid dissemination of critical information between platoons.

6.1.2 ETSI MEC-based Vehicular Computing

The simulation model introduced in section 4.2 supports vehicles modifying their mobility state, such as those exiting a parking lot. Future enhancements aim to improve this tool to support fully mobile vehicles, facilitating automated distribution of MEC applications in more dynamic environments, also depending on specific mobility characteristics. These extensions will require adjustments in the architecture design outlined in section 4.1, particularly in automating resource acquisition and release procedures to make the MEC system entirely self-managed, eliminating the need for device-initiated processes. Initial steps toward this goal were taken during this dissertation, which included developing a system-specific MEC App that monitors the AoI as defined by the MEC-H. Utilizing RNI services, this application tracks new devices entering and leaving the coverage area of the MEC-H-related 5G base station. This research highlighted different challenges, such as identifying during resource allocation vehicles staying enough time in the AoI for task execution and managing frequent migrations as vehicles move between network cells.

The architectural framework proposed in this dissertation provides a foundation for novel VC computing applications, including Federated Learning [160] environments, which are currently uncommon in those targeted by the community. Future work could explore deploying a MEC App as a federated server on the MEC-H local resource infrastructure, capable of coordinating remote MEC Apps (federated clients) during local model training. The federated server would select clients based on model descriptors and other information gathered via the MEC standard API, allowing the clients, upon receiving the model, to train on local data while utilizing their internal status, local data characteristics, and assigned reward mechanisms.

Furthermore, recent advancements in the ETSI MEC reference architecture [11, 161, 162] reflect a growing interest in enabling service continuity across multiple mobile network operators and infrastructure providers. The MEC federation model [161] aims to enable edge computing as an operator-agnostic service, allowing users to access edge applications across different operators' edge clouds. Future plans will explore this federation model to support the deployment and continuity of stateful vehicular applications, including inter-operator migration and cross-region roaming. These studies will be essential to fully realize MEC-enabled edge infrastructure that supports highly mobile vehicular contexts, where seamless and cross-regional service continuity has become crucial.

6.1.3 Vehicular Computing Through MEC and O-RAN Integration

As mentioned in section 5.1.2, two main challenges affect the integration of O-RAN and MEC standards. This dissertation focused on one of these challenges by creating a framework simplifying xApps development. The framework introduces an abstraction layer to reduce the complexity associated with interactions over the E2 interface, thus streamlining the development process. However, as discussed in section 5.2.5, this framework presents several challenges that require further exploration in future studies.

Furthermore, future research will focus on extending this framework to support the creation of MEC services using xApps, thereby enabling the deployment of a fully MEC-compliant edge node within an O-RAN-compliant 5G network.

Building on the integration of O-RAN and MEC standards, future work will focus on developing a real-world testbed of the architecture proposed in Chapter 4. Leveraging the reference architecture provided by the O-RAN alliance, its integration with the ETSI MEC standard, and innovative open-source frameworks for software-defined vehicles (e.g., SOAFEE [18]), it will possible to enable the deployment of MEC-compliant applications across both edge nodes and highly mobile UEs, such as vehicles, within a 5G network.

6.2 Concluding Remarks

This dissertation presented foundational solutions addressing critical challenges in vehicular communications and edge computing, paving the way for more robust and efficient next-generation vehicular applications. The key contributions of this work are summarized as follows:

- It thoroughly investigated two approaches to ensure security and to meet strict QoS requirements in VANET scenarios. This required the creation of a dataset containing V2V/V2I-based messages compliant with ETSI standard.

- It explored the possibility of extending MEC-compliant edge node resources with those included in far-edge devices, improving the computational scope of edge nodes.

- To evaluate this extension, a comprehensive simulation tool was developed and shared with the research community, allowing researchers to test applications within a VC context that complies with MEC standards.

- It studied application migration in dynamic vehicular environments, leveraging standard-compliant APIs to support service continuity in VC scenarios.

- Finally, it extensively investigated the deployment of MEC standard within a 5G network by exploiting the O-RAN specifications. This resulted in an openly available framework to support the development of O-RAN near real-time applications, representing a key enabler toward achieving this integration.

Bibliography

[1] T. Meng, J. Huang, C.-M. Chew, D. Yang, and Z. Zhong, "Configuration and design schemes of environmental sensing and vehicle computing systems for automated driving: A review," *IEEE Sensors Journal*, vol. 23, no. 14, pp. 15305–15320, 2023.

[2] S. Lu and W. Shi, "Vehicle computing: Vision and challenges," *Journal of Information and Intelligence*, vol. 1, no. 1, pp. 23–35, 2023.

[3] R. Meneguette, R. De Grande, J. Ueyama, G. P. R. Filho, and E. Madeira, "Vehicular edge computing: Architecture, resource management, security, and challenges," *ACM Comput. Surv.*, vol. 55, nov 2021.

[4] S. Olariu, "A survey of vehicular cloud research: Trends, applications and challenges," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 6, pp. 2648–2663, 2020.

[5] IEEE, "IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments," *IEEE Std 802.11p-2010 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, IEEE Std 802.11n-2009, and IEEE Std 802.11w-2009)*, pp. 1–51, 2010.

[6] J. Meredith, F. Firmin, and M. Pope, "Release 16 description; summary of rel-16 work items," tech. rep., 3GPP, 2022.

[7] L. Liu, C. Chen, Q. Pei, S. Maharjan, and Y. Zhang, "Vehicular edge computing and networking: A survey," *Mobile networks and applications*, vol. 26, pp. 1145–1168, 2021.

[8] T. ETSI, "Etsi tr 102 638-intelligent transport systems (its); vehicular communications; basic set of applications; release 2," *Vehicular Communications; Basic Set of Applications; Definitions*, vol. 1, 2024.

[9] F. Giust, V. Sciancalepore, D. Sabella, M. C. Filippou, S. Mangiante, W. Featherstone, and D. Munaretto, "Multi-access edge computing: The driver behind the wheel of 5g-connected cars," *IEEE Communications Standards Magazine*, vol. 2, no. 3, pp. 66–73, 2018.

[10] M. De Donno, K. Tange, and N. Dragoni, "Foundations and evolution of modern computing paradigms: Cloud, iot, edge, and fog," *IEEE Access*, vol. 7, pp. 150936–150948, 2019.

[11] ETSI, "GS MEC 003 - V3.2.1." `https://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/03.02.01_60/gs_mec003v030201p.pdf`, 4 2024.

[12] AECC, "Publications and use cases," Nov 2021.

[13] S. Bitam, A. Mellouk, and S. Zeadally, "Vanet-cloud: a generic cloud computing model for vehicular ad hoc networks," *IEEE Wireless Communications*, vol. 22, no. 1, pp. 96–102, 2015.

[14] M. Gerla, "Vehicular Cloud Computing," in *Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, pp. 152–155, 2012.

[15] S. Olariu, I. Khalil, and M. Abuelela, "Taking VANET to the Clouds," *International Journal of Pervasive Computing and Communications*, vol. 7, no. 1, pp. 7–21, 2011.

[16] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[17] T. Mekki, I. Jabri, A. Rachedi, and M. ben Jemaa, "Vehicular cloud networks: Challenges, architectures, and future directions," *Vehicular Communications*, vol. 9, pp. 268–280, 2017.

[18] S. SIG, "Scalable open architecture for embedded edge." `https://www.soafee.io/`, 9 2024.

[19] T. ETSI, "Intelligent Transport Systems (ITS);Vehicular Communications; Basic Set of Applications; Cooperative Awareness Service; Release 2," *ETSI TS*, vol. 20, 2023.

[20] E. ETSI, ""Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 3: Specification of Descentralized Environmental Notification Basic Service"," *IEEE Open*, 2020.

[21] T. ETSI, "Intelligent Transport Systems (ITS); Cooperative ITS (C-ITS);Release 1 ," *Technical Report ETSI TR*, no. 101 607, pp. 448–51, 2020.

[22] I. A. Sumra, I. Ahmad, H. Hasbullah, and J.-l. bin Ab Manan, "Classes of attacks in VANET," in *2011 Saudi International Electronics, Communications and Photonics Conference (SIECPC)*, pp. 1–5, 2011.

[23] F. Sakiz and S. Sen, "A survey of attacks and detection mechanisms on intelligent transportation systems: VANETs and IoV," *Ad Hoc Networks*, vol. 61, pp. 33–50, 2017.

[24] A. Bujari, M. Conti, C. De Francesco, and C. E. Palazzi, "Fast multi-hop broadcast of alert messages in VANETs: An analytical model," *Ad Hoc Networks*, vol. 82, pp. 126–133, 2019.

[25] Y. Jeon, S. Kuk, and H. Kim, "Reducing message collisions in sensing-based semi-persistent scheduling (sps) by using reselection lookaheads in cellular v2x," *Sensors*, vol. 18, no. 12, 2018.

[26] F. H. Rahman *et al.*, "Off-Street Vehicular Fog for Catering Applications in 5G/B5G: A Trust-Based Task Mapping Solution and Open Research Issues," *IEEE Access*, vol. 8, pp. 117218–117235, 2020.

[27] C. Ma *et al.*, "Parking Edge Computing: Parked-Vehicle-Assisted Task Offloading for Urban VANETs," *IEEE Internet of Things Journal*, vol. 8, no. 11, pp. 9344–9358, 2021.

[28] W. Fan, J. Liu, M. Hua, F. Wu, and Y. Liu, "Joint Task Offloading and Resource Allocation for Multi-Access Edge Computing Assisted by Parked and Moving Vehicles," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 5, pp. 5314–5330, 2022.

[29] Y. Peng, B. Shi, T. Jiang, X. Tu, D. Xu, and K. Hua, "A survey on in-vehicle time-sensitive networking," *IEEE Internet of Things Journal*, vol. 10, no. 16, pp. 14375–14396, 2023.

[30] A. Feraudo, A. Calvio, A. Bujari, and P. Bellavista, "A novel design for advanced 5g deployment environments with virtualized resources at vehicular and mec nodes," in *2023 IEEE Vehicular Networking Conference (VNC)*, pp. 97–103, IEEE, 2023.

[31] A. Feraudo, "Phd forum abstract: Vehicular-based support to cooperative edge computing based applications in next-gen networks," in *Proceedings of the 22nd International Conference on Information Processing in Sensor Networks*, pp. 352–353, 2023.

[32] A. Feraudo., A. Calvio., and P. Bellavista., "A novel omnet++-based simulation tool for vehicular cloud computing in etsi mec-compliant 5g environments," in *Proceedings of the 13th*

International Conference on Simulation and Modeling Methodologies, Technologies and Applications - SIMULTECH, pp. 448–455, INSTICC, SciTePress, 2023.

[33] A. Feraudo, A. Calvio, and P. Bellavista, "Simulating and Validating Vehicular Cloud Computing Applications in MEC-enabled 5G Environments," *Journal of Simulation Engineering*, vol. 4, pp. 9–1, 2024.

[34] A. Feraudo, N. Romandini, C. Mazzocca, R. Montanari, and P. Bellavista, "DIVA: A DID-based reputation system for secure transmission in VANETs using IOTA," *Computer Networks*, p. 110332, 2024.

[35] A. Feraudo, S. Maxenti, A. Lacava, P. Bellavista, M. Polese, and T. Melodia, "xDevSM: Streamlining xapp development with a flexible framework for o-ran e2 service models," in *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, ACM MobiCom '24, (New York, NY, USA), p. 1954–1961, Association for Computing Machinery, 2024.

[36] A. Feraudo, A. Garbugli, and P. Bellavista, "Controlling Communications Quality in V2V Platooning: a TSN-like Slot-Based Scheduler Approach," *arXiv preprint arXiv:2405.01301*, 2024.

[37] A. Feraudo, D. A. Popescu, P. Yadav, R. Mortier, and P. Bellavista, "Mitigating IoT botnet DDoS attacks through MUD and ebpf based traffic filtering," in *Proceedings of the 25th International Conference on Distributed Computing and Networking*, pp. 164–173, 2024.

[38] 3rd Generation Partnership Project, "3gpp tr 21.914 v14. 0.0 (2018-05) 3rd generation partnership project; technical specification group services and system aspects; release 14 description; summary of rel-14 work items (release 14)," tech. rep., 3GPP, 2018.

[39] B. Sun and H. Zhang, "802.11 NGV proposed PAR," in *Proc. IEEE NGV Meeting*, pp. 2–3, 2018.

[40] B. Y. Yacheur, T. Ahmed, and M. Mosbah, "Analysis and Comparison of IEEE 802.11p and IEEE 802.11bd," in *Communication Technologies for Vehicles* (F. Krief, H. Aniss, L. Mendiboure, S. Chaumette, and M. Berbineau, eds.), (Cham), pp. 55–65, Springer International Publishing, 2020.

[41] "IEEE Guide for Wireless Access in Vehicular Environments (WAVE) Architecture," *IEEE Std 1609.0-2019 (Revision of IEEE Std 1609.0-2013)*, pp. 1–106, 2019.

[42] E. ETSI, "ETSI EN 302 663 V1. 3.11, Intelligent Transport Systems (ITS); ITS-G5 Access layer specification for Intelligent Transport Systems operating in the 5 GHz frequency bands," *ITS-G5 Access layer specification for Intelligent Transport Systems operating in the 5 GHz frequency bands*, 2020.

[43] T. ETSI, "102 636-4-2 v1.4.1," *Intelligent transport systems (ITS); Geonetworking*, 2021.

[44] T. ETSI, "102 636-3 v1.1.1," *Intelligent Transport Systems (ITS);Vehicular Communications; GeoNetworking; Part 3: Network architecture*, 2010.

[45] T. ETSI, "101 556-1 v1. 1.1 (2012-07) intelligent transport systems (its)," *Infrastructure to Vehicle Communication;Electric Vehicle Charging Spot Notification Specification*, 2012.

[46] 3GPP, "Release 15 description;summary of rel-15 work items(release 15)," Tech. Rep. TR 21.915 V15.0.0 (2019-09), 3GPP, 2019.

[47] S. Kekki, W. Featherstone, Y. Fang, P. Kuure, A. Li, A. Ranjan, D. Purkayastha, F. Jiangping, D. Frydman, G. Verin, *et al.*, "Mec in 5g networks," *ETSI white paper*, vol. 28, no. 2018, pp. 1–28, 2018.

[48] O-RAN alliance. `https://www.o-ran.org/`, 2024.

[49] O-RAN Working Group 3, "O-RAN Near-Real-time RAN Intelligent Controller Architecture & E2 General Aspects and Principles 2.00." O-RAN.WG3.E2GAP-v02.01 Technical Specification, 7 2021.

[50] O-RAN Working Group 3, "O-RAN near-real-time RAN intelligent controller, E2 application protocol 2.00." O-RAN.WG3.E2AP-v02.00 Technical Specification, 7 2020.

[51] O-RAN Working Group 3, "O-RAN near-real-time RAN intelligent controller E2 service model 2.00." ORAN-WG3.E2SM-v02.00 Technical Specification, 7 2021.

[52] O-RAN Working Group 3, "O-RAN near-real-time RAN intelligent controller E2 service model (E2SM) KPM 2.0." ORAN-WG3.E2SM-KPM-v02.00 Technical Specification, 7 2021.

[53] O-RAN Working Group 3, "O-RAN near-real-time RAN intelligent controller E2 service model, ran control 1.0." ORAN-WG3.E2SM-RC-v01.00 Technical Specification, 7 2021.

[54] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding o-ran: Architecture, interfaces, algorithms, security, and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 2, pp. 1376–1411, 2023.

[55] O-RAN Software Community, "Near-RT RIC." `https://lf-o-ran-sc.atlassian.net/wiki/spaces/RICP/overview`. Accessed October 2024.

[56] R. Schmidt, M. Irazabal, and N. Nikaein, "Flexric: An sdk for next-generation sd-rans," in *Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies*, pp. 411–425, 2021.

[57] O-RAN Software Community, "RIC Applications (RICAPP)." `https://lf-o-ran-sc.atlassian.net/wiki/spaces/RICA/pages/13569056/Current+xApps`. Accessed July 2024.

[58] W. Recommendation, "Verifiable Credentials Data Model v1.1," *W3*, 2022.

[59] S. Popov, "The tangle," *White paper*, vol. 1, no. 3, p. 30, 2018.

[60] X. Li, T. Jing, R. Li, H. Li, X. Wang, and D. Shen, "BDRA: Blockchain and Decentralized Identifiers Assisted Secure Registration and Authentication for VANETs," *IEEE Internet of Things Journal*, pp. 1–1, 2022.

[61] C. P. Fernandes, C. Montez, D. D. Adriano, A. Boukerche, and M. S. Wangham, "A blockchain-based reputation system for trusted VANET nodes," *Ad Hoc Networks*, vol. 140, p. 103071, 2023.

[62] Z. Lu, Q. Wang, G. Qu, and Z. Liu, "BARS: A Blockchain-Based Anonymous Reputation System for Trust Management in VANETs," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pp. 98–103, 2018.

[63] Z. Yang, K. Yang, L. Lei, K. Zheng, and V. C. M. Leung, "Blockchain-Based Decentralized Trust Management in Vehicular Networks," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1495–1505, 2019.

[64] N. Li, Y. Guo, Y. Chen, and J. Chai, "A Partitioned DAG Distributed Ledger with Local Consistency for Vehicular Reputation Management," *Wireless Communications and Mobile Computing*, vol. 2022, p. 6833535, Mar 2022.

[65] Y. Li, X. Tao, X. Zhang, J. Xu, Y. Wang, and W. Xia, "A DAG-Based Reputation Mechanism for Preventing Peer Disclosure in SIoV," *IEEE Internet of Things Journal*, vol. 9, no. 23, pp. 24095–24106, 2022.

[66] G. Du, Y. Cao, J. Li, and Y. Zhuang, "Secure Information Sharing Approach for Internet of Vehicles Based on DAG-Enabled Blockchain," *Electronics*, vol. 12, no. 8, 2023.

[67] H. M. Song, J. Woo, and H. K. Kim, "In-vehicle network intrusion detection using deep convolutional neural network," *Vehicular Communications*, vol. 21, p. 100198, 2020.

[68] S. Jeong, S. Lee, H. Lee, and H. K. Kim, "X-canids: Signal-aware explainable intrusion detection system for controller area network-based in-vehicle network," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 3, pp. 3230–3246, 2024.

[69] M. L. Han, B. I. Kwak, and H. K. Kim, "Anomaly intrusion detection method for vehicular networks based on survival analysis," *Vehicular Communications*, vol. 14, pp. 52–63, 2018.

[70] M. E. Verma, R. A. Bridges, M. D. Iannacone, S. C. Hollifield, P. Moriano, S. C. Hespeler, B. Kay, and F. L. Combs, "A comprehensive guide to can ids data and introduction of the road dataset," *PLoS one*, vol. 19, no. 1, p. e0296879, 2024.

[71] J. Gozalvez, M. Sepulcre, and R. Bauza, "Ieee 802.11p vehicle to infrastructure communications in urban environments," *IEEE Communications Magazine*, vol. 50, no. 5, pp. 176–183, 2012.

[72] S. B. Cruz, T. E. Abrudan, Z. Xiao, N. Trigoni, and J. Barros, "Neighbor-aided localization in vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 10, pp. 2693–2702, 2017.

[73] S. Malebary, "DSRC Vehicle Communications." UCI Machine Learning Repository, 2016. DOI: https://doi.org/10.24432/C5R615.

[74] B. Kihei, "Ieee 802.11p wireless congestion and jamming experiments," 2022.

[75] M. Boban, C. Jiao, and M. Gharba, "Measurement-based evaluation of uplink throughput prediction," in *2022 IEEE 95th Vehicular Technology Conference: (VTC2022-Spring)*, pp. 1–6, 2022.

[76] R. Hernangómez, P. Geuer, A. Palaios, D. Schäufele, C. Watermann, K. Taleb-Bouhemadi, M. Parvini, A. Krause, S. Partani, C. Vielhaus, M. Kasparick, D. F. Külzer, F. Burmeister, F. H. P. Fitzek, H. D. Schotten, G. Fettweis, and S. Stańczak, "Berlin v2x: A machine learning dataset from multiple vehicles and radio access technologies," in *2023 IEEE 97th Vehicular Technology Conference (VTC2023-Spring)*, pp. 1–5, 2023.

[77] I. ETSI, "Intelligent transport systems (ITS); security; pre-standardization study on pseudonym change management," *Technical Report ETSI TR 103 415 V1. 1.1 (2018-04)*, 2018.

[78] V. Martinez and F. Berens, "Survey on its-g5 cam statistics," tech. rep., CAR 2 CAR Communication Consortium, 12 2018.

[79] C. Obermaier, R. Riebl, and C. Facchi, "Dynamic scenario control for VANET simulations," in *2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, pp. 681–686, 2017.

[80] W. Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, and A. Ahmed, "Edge computing: A survey," *Future Generation Computer Systems*, vol. 97, pp. 219–235, 2019.

[81] R. Riebl, H.-J. Günther, C. Facchi, and L. Wolf, "Artery: Extending veins for vanet applications," in *2015 International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, pp. 450–456, 2015.

[82] A. Hegde and A. Festag, "Artery-C: An OMNeT++ based discrete event simulation framework for cellular V2X," in *Proceedings of the 23rd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pp. 47–51, 2020.

[83] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wiessner, "Microscopic Traffic Simulation using SUMO," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2575–2582, 2018.

[84] C. . CAR, "Car 2 car communication consortium triggering conditions and data quality traffic jam car 2 car communication consortium about the c2c-cc," *white paper*, 2022.

[85] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198–208, 1983.

[86] M. Safwat, A. Elgammal, E. G. AbdAllah, and M. A. Azer, "Survey and taxonomy of information-centric vehicular networking security attacks," *Ad Hoc Networks*, vol. 124, p. 102696, 2022.

[87] S. Mazhar, A. Rakib, L. Pan, F. Jiang, A. Anwar, R. Doss, and J. Bryans, "State-of-the-art authentication and verification schemes in vanets: A survey," *Vehicular Communications*, vol. 49, p. 100804, 2024.

[88] N.-W. Lo and H.-C. Tsai, "Illusion attack on vanet applications-a message plausibility problem," in *2007 IEEE globecom workshops*, pp. 1–8, IEEE, 2007.

[89] J. Zang, V. Towhidlou, and M. Shikh-Bahaei, "Collision avoidance in v2x communication networks," in *2019 IEEE Wireless Communications and Networking Conference Workshop (WCNCW)*, pp. 1–6, 2019.

[90] T. Maruko, S. Yasukawa, R. Kudo, S. Nagata, and M. Iwamura, "Packet collision reduction scheme for lte v2x sidelink communications," in *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, pp. 1–5, 2018.

[91] A. Nasrallah, A. S. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, and H. ElBakoury, "Ultra-low latency (ull) networks: The ieee tsn and ietf detnet standards and related 5g ull research," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 88–145, 2018.

[92] T. Adame, M. Carrascosa-Zamacois, and B. Bellalta, "Time-sensitive networking in ieee 802.11 be: On the way to low-latency wifi 7," *Sensors*, vol. 21, no. 15, p. 4954, 2021.

[93] "Ieee standard for local and metropolitan area networks–timing and synchronization for time-sensitive applications," *IEEE Std 802.1AS-2020 (Revision of IEEE Std 802.1AS-2011)*, pp. 1–421, 2020.

[94] A. Garbugli, A. Bujari, and P. Bellavista, "End-to-end qos management in self-configuring tsn networks," in *2021 17th IEEE International conference on factory communication systems (WFCS)*, pp. 131–134, IEEE, 2021.

[95] Z. Satka, M. Ashjaei, H. Fotouhi, M. Daneshtalab, M. Sjödin, and S. Mubeen, "A comprehensive systematic review of integration of time sensitive networking and 5g communication," *Journal of systems architecture*, vol. 138, p. 102852, 2023.

[96] R. Reddy G and R. R, "An empirical study on mac layer in ieee 802.11p/wave based vehicular ad hoc networks," *Procedia Computer Science*, vol. 143, pp. 720–727, 2018. 8th International Conference on Advances in Computing & Communications (ICACC-2018).

[97] F. Borgonovo, A. Capone, M. Cesana, L. Fratta, *et al.*, "Rr-aloha, a reliable r-aloha broadcast channel for ad-hoc inter-vehicle communication networks," in *Proceedings of Med-Hoc-Net*, vol. 2002, Citeseer, 2002.

[98] R. Scopigno and H. A. Cozzetti, "Mobile slotted aloha for vanets," in *2009 IEEE 70th Vehicular Technology Conference Fall*, pp. 1–5, IEEE, 2009.

[99] H. A. Cozzetti and R. Scopigno, "Rr-aloha+: A slotted and distributed mac protocol for vehicular communications," in *2009 IEEE Vehicular Networking Conference (VNC)*, pp. 1–8, IEEE, 2009.

[100] Y. Kim, M. Lee, and T.-J. Lee, "Coordinated multichannel mac protocol for vehicular ad hoc networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 8, pp. 6508–6517, 2015.

[101] L. K. Ouladdjedid, M. B. Yagoubi, and H. Menouar, "Cssa mac: Carrier sense with slotted-aloha multiple access mac in vehicular network," *International Journal of Vehicle Information and Communication Systems*, vol. 3, no. 4, pp. 336–354, 2018.

[102] L. Lai, Z. Song, and W. Xu, "A novel framed slotted aloha medium access control protocol based on capture effect in vehicular ad hoc networks," *Sensors*, vol. 24, no. 3, p. 992, 2024.

[103] D. Jia, K. Lu, J. Wang, X. Zhang, and X. Shen, "A survey on platoon-based vehicular cyber-physical systems," *IEEE communications surveys & tutorials*, vol. 18, no. 1, pp. 263–284, 2015.

[104] S. Zeadally, J. Guerrero, and J. Contreras, "A tutorial survey on vehicle-to-vehicle communications," *Telecommunication Systems*, vol. 73, pp. 469–489, 2020.

[105] P. Fernandes and U. Nunes, "Platooning with ivc-enabled autonomous vehicles: Strategies to mitigate communication delays, improve safety and traffic flow," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 1, pp. 91–106, 2012.

[106] M. Segata, B. Bloessl, S. Joerer, C. Sommer, M. Gerla, R. L. Cigno, and F. Dressler, "Towards inter-vehicle communication strategies for platooning support," in *2014 7th International Workshop on Communication Technologies for Vehicles (Nets4Cars-Fall)*, pp. 1–6, IEEE, 2014.

[107] Y. Park and H. Kim, "Collision control of periodic safety messages with strict messaging frequency requirements," *IEEE transactions on vehicular technology*, vol. 62, no. 2, pp. 843–852, 2012.

[108] L. Cao, S. Roy, and H. Yin, "Resource allocation in 5g platoon communication: Modeling, analysis and optimization," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 4, pp. 5035–5048, 2022.

[109] H. Baniabdelghany, R. Obermaisser, and A. Khalifeh, "Extended synchronization protocol based on ieee802.1as for improved precision in dynamic and asymmetric tsn hybrid networks," in *2020 9th Mediterranean Conference on Embedded Computing (MECO)*, pp. 1–8, 2020.

[110] K. Alemdar, D. Varshney, S. Mohanti, U. Muncuk, and K. Chowdhury, "Rfclock: timing, phase and frequency synchronization for distributed wireless networks," in *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, MobiCom '21, (New York, NY, USA), p. 15–27, Association for Computing Machinery, 2021.

[111] A. Garg, A. Yadav, A. Sikora, and A. S. Sairam, "Wireless precision time protocol," *IEEE Communications Letters*, vol. 22, no. 4, pp. 812–815, 2018.

[112] K. F. Hasan, Y. Feng, and Y.-C. Tian, "Gnss time synchronization in vehicular ad-hoc networks: Benefits and feasibility," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 12, pp. 3915–3924, 2018.

[113] V. Martinez and F. Berens, "Survey on its-g5 cam statistics," tech. rep., CAR 2 CAR Communication Consortium, 12 2018.

[114] R. Molina-Masegosa, M. Sepulcre, J. Gozalvez, F. Berens, and V. Martinez, "Empirical models for the realistic generation of cooperative awareness messages in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 5713–5717, 2020.

[115] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[116] N. Liu, M. Liu, W. Lou, G. Chen, and J. Cao, "Pva in vanets: Stopped cars are not silent," in *2011 Proceedings IEEE INFOCOM*, pp. 431–435, 2011.

[117] P. Qin *et al.*, "Energy-Efficient Resource Allocation for Parked-Cars-Based Cellular-V2V Heterogeneous Networks," *IEEE Internet of Things Journal*, vol. 9, no. 4, pp. 3046–3061, 2022.

[118] F. Dressler, P. Handle, and C. Sommer, "Towards a vehicular cloud-using parked vehicles as a temporary network and storage infrastructure," in *Proceedings of the 2014 ACM international workshop on Wireless and mobile technologies for smart cities*, pp. 11–18, 2014.

[119] J. Feng, Z. Liu, C. Wu, and Y. Ji, "Ave: Autonomous vehicular edge computing framework with aco-based scheduling," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 12, pp. 10660–10675, 2017.

[120] X. Huang, R. Yu, J. Liu, and L. Shu, "Parked Vehicle Edge Computing: Exploiting Opportunistic Resources for Distributed Mobile Applications," *IEEE Access*, vol. 6, pp. 66649–66663, 2018.

[121] S. Kamakshi and V. Shankar Sriram, "Modularity based mobility aware community detection algorithm for broadcast storm mitigation in vanets," *Ad Hoc Networks*, vol. 104, p. 102161, 2020.

[122] M. S. Bute, P. Fan, G. Liu, F. Abbas, and Z. Ding, "A cluster-based cooperative computation offloading scheme for c-v2x networks," *Ad Hoc Networks*, vol. 132, p. 102862, 2022.

[123] C. Li *et al.*, "Parked Vehicular Computing for Energy-Efficient Internet of Vehicles: A Contract Theoretic Approach," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6079–6088, 2019.

[124] N. Cha, C. Wu, T. Yoshinaga, Y. Ji, and K.-L. A. Yau, "Virtual Edge: Exploring Computation Offloading in Collaborative Vehicular Edge Computing," *IEEE Access*, vol. 9, pp. 37739–37751, 2021.

[125] ETSI, "GS MEC 021 - V3.1.1," 2 2024.

[126] "OMNeT++ Event Discrete Simulator."

[127] G. Nardini, D. Sabella, G. Stea, P. Thakkar, and A. Virdis, "Simu5G–An OMNeT++ Library for End-to-End Performance Evaluation of 5G Networks," *IEEE Access*, vol. 8, pp. 181176 – 181191, 2020.

[128] N. S. Rajput, U. Singh, A. Dua, N. Kumar, J. J. P. C. Rodrigues, S. Sisodia, M. Elhoseny, and Y. Lakys, "Amalgamating vehicular networks with vehicular clouds, ai, and big data for next-generation its services," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–15, 2023.

[129] K. Hayawi, Z. Anwar, A. W. Malik, and Z. Trabelsi, "Airborne computing: A toolkit for uav-assisted federated computing for sustainable smart cities," *IEEE Internet of Things Journal*, vol. 10, no. 21, pp. 18941–18950, 2023.

[130] O. U. Akgül, W. Mao, B. Cho, and Y. Xiao, "Vfogsim: A data-driven platform for simulating vehicular fog computing environment," *IEEE Systems Journal*, vol. 17, no. 3, pp. 5002–5013, 2023.

[131] Z. Wei, C. Huang, B. Li, Y. Zhao, X. Cheng, L. Yang, and R. Zhang, "Airfogsim: A lightweight and modular simulator for uav-integrated vehicular fog computing," *arXiv preprint arXiv:2409.02518*, 2024.

[132] S. Ge, M. Cheng, X. He, and X. Zhou, "A two-stage service migration algorithm in parked vehicle edge computing for internet of things," *Sensors*, vol. 20, no. 10, 2020.

[133] T. Taleb, A. Ksentini, and P. A. Frangoudis, "Follow-me cloud: When cloud services follow mobile users," *IEEE Transactions on Cloud Computing*, vol. 7, no. 2, pp. 369–382, 2019.

[134] Q.-V. Pham, F. Fang, V. N. Ha, M. J. Piran, M. Le, L. B. Le, W.-J. Hwang, and Z. Ding, "A survey of multi-access edge computing in 5g and beyond: Fundamentals, technology integration, and state-of-the-art," *IEEE Access*, vol. 8, pp. 116974–117017, 2020.

[135] L. Tomaszewski, S. Kukliński, and R. Kołakowski, "A new approach to 5g and mec integration," in *Artificial Intelligence Applications and Innovations. AIAI 2020 IFIP WG 12.5 International Workshops: MHDW 2020 and 5G-PINE 2020, Neos Marmaras, Greece, June 5–7, 2020, Proceedings 16*, pp. 15–24, Springer, 2020.

[136] R. Xavier, R. S. Silva, M. Ribeiro, W. Moreira, L. Freitas, and A. Oliveira-Jr, "Integrating multi-access edge computing (mec) into open 5g core," in *Telecom*, vol. 5, pp. 433–450, MDPI, 2024.

[137] P. V. Wadatkar, R. G. Garroppo, and G. Nencioni, "5g-mec testbeds for v2x applications," *Future Internet*, vol. 15, no. 5, 2023.

[138] S.-C. Huang, Y.-C. Luo, B.-L. Chen, Y.-C. Chung, and J. Chou, "Application-aware traffic redirection: A mobile edge computing implementation toward future 5g networks," in *2017 IEEE 7th International Symposium on Cloud and Service Computing (SC2)*, pp. 17–23, 2017.

[139] S. Kukliński, L. Tomaszewski, and R. Kołakowski, "On o-ran, mec, son and network slicing integration," in *2020 IEEE Globecom Workshops (GC Wkshps*, pp. 1–6, 2020.

[140] ETSI, "GS MEC 012 - V2.2.1," 2 2022.

[141] ETSI, "GS MEC 030 - V3.3.1," 6 2024.

[142] M. A. Habibi, G. M. Yilma, X. Costa-Pérez, and H. D. Schotten, "Unifying 3gpp, etsi, and o-ran smo interfaces: Enabling slice subnets interoperability," in *2023 IEEE Future Networks World Forum (FNWF)*, pp. 1–8, 2023.

[143] M. Tsampazi, S. D'Oro, M. Polese, L. Bonati, G. Poitau, M. Healy, M. Alavirad, and T. Melodia, "PandORA: Automated Design and Comprehensive Evaluation of Deep Reinforcement Learning Agents for Open RAN," *arXiv preprint arXiv:2407.11747*, 2024.

[144] C. Puligheddu, J. Ashdown, C. F. Chiasserini, and F. Restuccia, "SEM-O-RAN: Semantic and Flexible O-RAN Slicing for NextG Edge-Assisted Mobile Systems," in *Proceedings of IEEE INFOCOM 2023*, (New York Area, NJ, USA), 5 2023.

[145] M. Zangooei, M. Golkarifard, M. Rouili, N. Saha, and R. Boutaba, "Flexible RAN Slicing in Open RAN With Constrained Multi-Agent Reinforcement Learning," *IEEE Journal on Selected Areas in Communications*, vol. 42, pp. 280–294, 2 2024.

[146] M. Irazabal and N. Nikaein, "TC-RAN: A Programmable Traffic Control Service Model for 5G/6G SD-RAN," *IEEE Journal on Selected Areas in Communications*, vol. 42, pp. 406–419, 2 2024.

[147] A. Lacava, M. Polese, R. Sivaraj, R. Soundrarajan, B. S. Bhati, T. Singh, T. Zugno, F. Cuomo, and T. Melodia, "Programmable and Customized Intelligence for Traffic Steering in 5G Networks Using Open RAN Architectures," *IEEE Transactions on Mobile Computing*, vol. 23, pp. 2882–2897, 4 2024.

[148] E. Coronado, S. Siddiqui, and R. Riggio, "Roadrunner: O-RAN-based cell selection in beyond 5G networks," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–7, IEEE, 2022.

[149] R. Smith, C. Freeberg, T. Machacek, and V. Ramaswamy, "An O-RAN approach to spectrum sharing between commercial 5G and government satellite systems," in *MILCOM 2021-2021 IEEE Military Communications Conference (MILCOM)*, pp. 739–744, IEEE, 2021.

[150] C.-C. Chen, M. Irazabal, C.-Y. Chang, A. Mohammadi, and N. Nikaein, "FlexApp: Flexible and Low-Latency xApp Framework for RAN Intelligent Controller," in *ICC 2023 - IEEE International Conference on Communications*, pp. 5450–5456, 2023.

[151] E. Moro, M. Polese, A. Capone, and T. Melodia, "An Open RAN Framework for the Dynamic Control of 5G Service Level Agreements," in *2023 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pp. 141–146, 2023.

[152] D. Villa, I. Khan, F. Kaltenberger, N. Hedberg, R. S. da Silva, S. Maxenti, L. Bonati, A. Kelkar, C. Dick, E. Baena, J. M. Jornet, T. Melodia, M. Polese, and D. Koutsonikolas,

"X5G: An Open, Programmable, Multi-vendor, End-to-end, Private 5G O-RAN Testbed with NVIDIA ARC and OpenAirInterface," 2024.

[153] S. Popić, D. Pezer, B. Mrazovac, and N. Teslić, "Performance evaluation of using Protocol Buffers in the Internet of Things communication," in *2016 International Conference on Smart Systems and Technologies (SST)*, pp. 261–265, 2016.

[154] O-RAN Software Community, "xapp-frame-py." `https://docs.o-ran-sc.org/projects/o-ran-sc-ric-plt-xapp-frame-py/en/latest/overview.html`, 2024.

[155] O-RAN Software Community. `https://wiki.o-ran-sc.org/display/ORANSDK/Overview`, 2024.

[156] srsRAN, "Open RAN Near-RT RIC and xApp framework," 2023. Accessed: 2024-07-10.

[157] O-RAN Software Community, "ric-app-kpimon-go repository." `https://github.com/o-ran-sc/ric-app-kpimon-go`, 2024. Accessed July 2024.

[158] O-RAN Software Community, "ric-app-bouncer repository." `https://github.com/o-ran-sc/ric-app-bouncer`, 2024. Accessed July 2024.

[159] O-RAN Working Group 3, "O-RAN near-real-time RAN intelligent controller E2 service model (E2SM) KPM 3.0." O-RAN.WG3.E2SM-KPM-R003-v03.00 Technical Specification, 3 2023.

[160] K. Bonawitz, "Towards federated learning at scale: Syste m design," *arXiv preprint arXiv:1902.01046*, 2019.

[161] ETSI, "GS MEC 040 - V3.2.1," 3 2024.

[162] M. Suzuki, T. Joh, H. Lee, W. Featherstone, N. Sprecher, D. Sabella, N. Oliver, S. Shailendra, F. Granelli, C. Costa, *et al.*, "Mec federation: Deployment considerations," *ETSI White Paper*, no. 49, 2022.