Dipartimento Ingegneria dell'Informazione (DII)
Dottorato di ricerca in ingegneria dell'informazione

IINF-01/A Electronics
CICLO XXXVII

Innovative solutions for Wireless WAN:

from Cellular communications for Platooning to LoRaWAN software defined nodes

Hossein Khalilnasl

Supervisor: Prof. Emiliano Sisinni

To my beloved papa.

Abstract

The rapid development of wireless communication technologies has paved the way for enhanced network functionalities in diverse applications, from IoT-based networks to autonomous vehicle platooning. This thesis investigates innovative approaches to improve wireless communication systems in two critical domains: vehicular networks and Internet of Things (IoT) infrastructures. The first part of the research focuses on improving Cellular Vehicle-to-Everything (C-V2X) communication for vehicle platooning. Specifically, it addresses the limitations of the Sensing Based Semi Persistent Scheduling (SB-SPS) algorithm in Out-of-Coverage (OoC) scenarios by introducing a dual-strategy mechanism. This approach integrates precise time synchronization-using beaconing from a platoon leader-and a partition-based scheduling scheme to minimize resource collisions. The simulation results demonstrate improved packet delivery reliability and reduced update delays, supporting safer and more efficient platoon operations even in congested traffic environments.

The second part of the thesis explores a novel architecture for LoRaWAN end nodes through the implementation of container-based virtualization and Software-Defined Radio (SDR) technologies. By decomposing the LoRaWAN protocol stack into microservices and deploying them using Docker containers, the proposed solution enables full virtualization of node functionalities, including the radio layer. Extensive real-world experiments and performance evaluations reveal that this architecture maintains communication reliability while significantly improving system flexibility, modularity, and resource efficiency. Comparisons between MATLAB- and C-based SDR implementations further highlight the benefits of optimized software design.

Together, these contributions propose forward-looking solutions for robust, scalable, and programmable wireless systems that align with the needs of future autonomous mobility and dynamic IoT networks.

Sommario

Lo sviluppo rapido delle tecnologie di comunicazione wireless ha aperto la strada a funzionalità avanzate delle reti in diverse applicazioni, dalle reti basate su IoT alla formazione di veicoli autonomi. Questa tesi esamina approcci innovativi per migliorare i sistemi di comunicazione wireless in due ambiti critici: le reti veicolari e le infrastrutture IoT. La prima parte della ricerca si concentra sul miglioramento della comunicazione C-V2X per il platooning dei veicoli. In particolare, affronta le limitazioni dell'algoritmo SB-SPS in scenari di fuori copertura (OoC), introducendo un meccanismo a doppia strategia. Questo approccio integra una sincronizzazione temporale precisa, utilizzando il beaconing da parte di un veicolo leader del platoon, e uno schema di pianificazione basato su partizioni per ridurre al minimo le collisioni di risorse. I risultati delle simulazioni dimostrano un miglioramento nell'affidabilità della consegna dei pacchetti e una riduzione dei ritardi di aggiornamento, a supporto di operazioni più sicure ed efficienti del platoon anche in ambienti di traffico congestionato.

La seconda parte della tesi esplora una nuova architettura per i nodi finali LoRaWAN attraverso l'implementazione di virtualizzazione basata su container e tecnologie SDR. Decomponendo lo stack del protocollo LoRaWAN in microservizi e distribuendoli utilizzando container Docker, la soluzione proposta consente la virtualizzazione completa delle funzionalità del nodo, incluso il livello radio. Esperimenti pratici e valutazioni delle prestazioni mostrano che questa architettura mantiene l'affidabilità della comunicazione, migliorando significativamente la flessibilità, la modularità e l'efficienza delle risorse del sistema. I confronti tra implementazioni SDR basate su MATLAB e C evidenziano ulteriormente i vantaggi di un design software ottimizzato.

Complessivamente, queste contribuzioni propongono soluzioni innovative per sistemi wireless robusti, scalabili e programmabili, che rispondono alle esigenze della mobilità autonoma futura e delle reti IoT dinamiche.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my PhD supervisor, Prof. Emiliano Sisinni, for his invaluable guidance, continuous support, and encouragement throughout the course of my doctoral studies. I am also sincerely thankful to the PhD coordinator, Prof. Ivano Alessandri, for his support and for fostering a stimulating academic environment. A heartfelt thank you goes to Marzieh Bastanfar, who has been by my side during this journey, offering constant help and motivation. I am truly grateful to my friends for the laughter, good times, and the moments of respite they brought into my life. Finally, and most importantly, I want to express my deep love and appreciation to my family, whose unwavering support, patience, and affection have been my greatest source of strength.

Contents

List of Figures

List of Tables

Terminology

Abbreviations and Acronyms

3GPP	3rd Generation Partnership Project
5GAA	5G Automotive Association
ABP	Activation by Personalization
ACC	Adaptive Cruise Control
ADAS	Advanced Driver Assistance Systems
ADR	Adaptive Data Rate
AppSKey	Application Session Key
AES	Advanced Encryption Standard
CA	Cooperative Awareness
CAM	Cooperative Awareness Message
CAV	Connected and Automated Vehicle
C-ITS	Cooperative Intelligent Transport Systems
CBR	Channel Busy Ratio
CoAP	Constrained Application Protocol
CoCoCo	Communications-Control Co-Design
CR	Coding Rate
CRR	Coordinating Resource Reservation
CSS	Chirp Spread Spectrum
CSR	Candidate Single-Subframe Resource
DEN	Dynamic Event Notification
DENM	Decentralized Environmental Notification Message
D2D	Device-to-Device
DR	Data Rate
ETSI	European Telecommunications Standards Institute
eV2X	enhanced Vehicle-to-Everything
FS-CSS	Frequency Shift - Chirp Spread Spectrum
FEC	Forward Error Correction
FPGA	Field-Programmable Gate Array
GRaSPS	Gapped Reservation-Augmented SPS

GW	Gateway
GWs	Gateways
HD	Half-Duplex
HDPL	High-Density Platooning
IFTs	Information Flow Topologies
IIO	Industrial Input/Output
IETF	Internet Engineering Task Force
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
JS	Join Server
KVM	Kernel-based Virtual Machine
LIDAR	Light Detection And Ranging
LPWAN	Low Power Wide Area Networks
LOS	Line-of-sight
LXC	Linux Container
MAC	Medium Access Control
M2M	machine-to-machine
MCS	Modulation and Coding Scheme
MHDR	MAC Header
NB-IoT	Narrowband Internet of Things
NAS	Network Attached Storage
NS	Network Server
OTAA	Over-The-Air-Activation
OoC	Out-of-Coverage
PCA-RS	Packet Collision Avoidance Resource Selection
PDR	Packet Delivery Ratio
PRR	Packet Reception Ratio
PSCCH	Physical Sidelink Control Channel
PSSCH	Physical Sidelink Shared Channel
QoS	Quality of Service
RC	Resource Reselection Counter
RRI	Resource Reservation Interval
RSSI	Sidelink Reference Signal Strength Indicator
RSRP	Reference Signal Received Power
RSU	Roadside Unit
RSUs	Road Side Units
SCI	Sidelink Control Information
SF	Spreading Factor
SB-SPS	Sensing Based Semi Persistent Scheduling
STS-RS	Short Term Sensing-Based Resource Selection
SDN	Software-Defined Networking
SDR	Software-Defined Radio
SF	Spreading Factor

SBI	Southbound Interface
SR	Symbol Rate
TB	Transport Block
TP	Transmission Power
TTN	The Things Network
V2I	Vehicle-to-Infrastructure
V2N	Vehicle-to-Network
V2P	Vehicle-to-Pedestrian
V2X	Vehicle-to-Everything
V2V	Vehicle-to-Vehicle
VUE	Vehicular User Equipment
WAVE	Wireless Access in Vehicular Environment
WBSs	Wireless Blind Spots
WWAN	Wireless Wide Area Network

Chapter 1

Introduction

1.1 Background

The evolution of wireless communication technologies has introduced transformative solutions for both vehicular communication and IoT networks. Among the various emerging technologies, C-V2X and LoRaWAN stand out as critical enablers of next-generation wireless networks. These technologies address the increasing demands for connectivity in both autonomous transportation and large-scale IoT deployments.

C-V2X: The rapid evolution of vehicular communication technologies has introduced C-V2X, a crucial technology aimed at enabling seamless communication between vehicles. At its core, C-V2X utilizes direct communication modes through PC5 interfaces, defined in standards LTE-V2X and NR-V2X, facilitating reliable communication even in environments without direct network coverage (out-of-coverage, OoC). Among various applications, vehicle platooning-where vehicles travel in coordinated formations, closely spaced for improved safety, reduced fuel consumption, and increased road throughput-stands out as a prominent use case. However, standard resource allocation methods, specifically the SB-SPS scheme in LTE-V2X, exhibit critical vulnerabilities, notably in terms of collision probability and communication reliability due to unsynchronized resource selection and random resource allocation processes. Enhancing this technology, therefore, necessitates novel strategies that effectively mitigate these weaknesses, particularly focusing on time synchronization and partition-based scheduling mechanisms to improve platooning communication reliability and efficiency.

LoRaWAN and Virtualization: LoRaWAN is a widely adopted protocol within the domain of Low Power Wide Area Networks (LPWAN), specifically designed to facilitate IoT applications characterized by long-range and low-power communication requirements. The conventional LoRaWAN architecture typically comprises end devices, gateways, and backend infrastructure. Traditional implementations tightly couple node functionalities with specialized hardware, resulting in limited flexibility

and scalability in dynamic IoT deployments.

Addressing these limitations, recent advances propose a software-defined and virtualization-based approach, utilizing Linux Container (LXC), notably through lightweight Docker virtualization technologies. This method decomposes the traditional LoRaWAN node stack into discrete, containerized services, encompassing application, LoRaWAN protocol management, and the LoRa physical layer functionalities. Leveraging SDR, this virtualization extends even to the physical communication layer, replacing conventional hardware radios with fully virtualized alternatives. Such a comprehensive virtualization not only increases the flexibility and scalability of LoRaWAN deployments but also significantly enhances network management capabilities by facilitating dynamic configuration and rapid deployment of IoT solutions.

Thus, this thesis investigates and evaluates the practical performance and feasibility of fully virtualized LoRaWAN nodes, analyzing critical metrics such as latency and resource utilization under different virtualization scenarios and software implementations. The outcomes of this research contribute significantly to future wireless network management practices, providing insights into optimized deployments and resource efficiency in IoT applications.

1.2 Research Motivation

This thesis is motivated by the emerging need for scalable, energy-efficient, and flexible communication solutions in both vehicular networks and IoT infrastructures. Two primary goals drive the research:

Goal 1: *Improve the reliability and efficiency of C-V2X communication for vehicle platooning*

Vehicle platooning -a key use case in autonomous transportation- requires reliable communication in both in-coverage and out-of-coverage scenarios. The current SB-SPS mechanism suffers from a collision problem in Mode 4 operations. These deficiencies lead to degraded performance in safety-critical applications. The first goal is to address these shortcomings by integrating beacon-based time synchronization and a partition-based resource scheduling approach. This approach aims to:

- Minimize resource collisions,
- Reduce update delays, and
- Improve communication reliability and coordination across platoon members.

Goal 2: *Enable flexible and resource-efficient IoT deployments via LoRaWAN virtualization*

Conventional LoRaWAN implementations bind node functions to specific hardware, which limits flexibility and scalability. The second goal of this research is

to develop and evaluate a fully virtualized LoRaWAN end-node architecture using container-based virtualization and SDR. This goal encompasses:

- Decoupling LoRaWAN node functions into modular services,
- Enhancing deployment flexibility and manageability, and
- Optimizing performance in terms of latency and resource utilization.

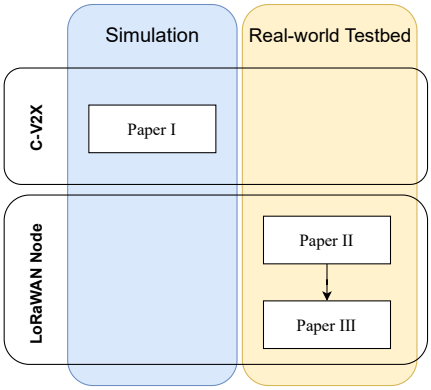


Figure 1.1: Mapping of the publications to the research goals

The mapping of the publications to the research goals is shown in Fig. 1.1. Paper I focuses on *Goal 1*, while Papers II and III contribute to *Goal 2*. The figure also indicates the methodologies employed for each publication, highlighting the use of both simulation and real-world testbed approaches to achieve the respective research objectives. By accomplishing these goals, this thesis aims to introduce innovative strategies and architectural solutions to address the evolving demands of next-generation wireless networks.

1.3 Research Questions

From the research goals, two separate sets of research questions were formulated pertaining to C-V2X and LoRaWAN.

Research Question 1

How can the reliability of C-V2X communications be improved for vehicle platooning in out-of-coverage scenarios?

C-V2X communications, especially in OoC scenarios where network infrastructure is unavailable, rely heavily on sidelink communication through the PC5 interface. In such cases, the SB-SPS scheme of LTE-V2X Mode 4 or NR-V2X Mode 2 is responsible for resource allocation. However, SB-SPS is known to suffer from random resource selection and lack of coordination in time and frequency domains, leading to increased interference and packet collisions. These drawbacks pose a significant risk to the reliability of safety-critical applications like vehicle platooning, where precise and timely message exchange among vehicles is vital for maintaining formation and avoiding accidents.

Research Question 2

Is it feasible to implement a fully software-defined LoRaWAN end node using lightweight container-based virtualization?

This question investigates whether lightweight virtualization technologies, such as LXC, can effectively support a fully software-defined LoRaWAN end node. The feasibility here refers not only to the functional correctness of the design but also to its performance, modularity, and practicality in real-world deployments. In the proposed architecture, each functional block of the LoRaWAN stack is implemented as a separate microservice running in its own container. This allows for clear separation of concerns, easier maintenance, and deployment flexibility. The study evaluates this setup in a real-world testbed, using a software-defined radio (SDR) as the physical layer interface. While the results confirm that the approach is technically viable, they also identify a key bottleneck in the data throughput between the SDR and the host system. This highlights that while containerization is promising for creating flexible and maintainable LoRaWAN nodes, achieving optimal performance requires addressing infrastructure limitations, particularly in data handling between virtualized components and physical interfaces.

Research Question 3

Can container-based virtualization be effectively used to virtualize LoRaWAN end nodes while maintaining performance and scalability in IoT deployments?

This research question examines whether lightweight container technologies, such as Docker LXCs, can support the virtualization of LoRaWAN end nodes in a way that is not only functionally viable but also efficient and scalable for IoT applications. The idea builds on the principles of modularity and abstraction-where each layer or function of a LoRaWAN node (e.g., application logic, MAC protocol handling, radio interface) can be isolated into individual, reusable microservices. Such virtualization could offer significant advantages in terms of deployment flexibility, ease of maintenance, scalability, and adaptability to diverse hardware platforms. However, this also raises concerns about whether the performance-especially in terms of latency, reliability, and resource usage-can be preserved in a constrained environment typical of IoT applications. The question thus centers on balancing the benefits of

modular software design with the strict performance requirements expected from real-world, low-power wireless networks.

1.4 Thesis Contributions

The primary contributions of this thesis can be summarized as follows:

1. **Enhanced C-V2X Platooning Communication (Paper I):**

   To answer *Research Question 1*, we proposed a novel dual-strategy communication scheme aimed at improving the reliability of C-V2X communications for vehicle platooning in out-of-coverage scenarios. The purpose is to overcome the limitations of the existing SB-SPS mechanism by introducing better coordination in both time and resource domains. The proposed contributions can be summarized as follows:

   - Introduction of a beacon-based time synchronization method to align Cooperative Awareness Message (CAM) generation among platoon members.
   - Design and implementation of a partition-based MAC-layer resource scheduling scheme to minimize random resource contention and transmission collisions.

   This work's results show that the combination of time synchronization and partitioned scheduling significantly enhances the reliability and predictability of V2V communication in platooning scenarios, reducing the risk of packet losses and improving overall platoon stability in out-of-coverage environments.

2. **Fully Virtualized LoRaWAN Node Implementation (Paper II):**

   To answer *Research Question 2*, we proposed the implementation of a fully software-defined LoRaWAN end node, realized through a combination of container-based virtualization. The purpose is to explore the feasibility of decomposing a LoRaWAN node into modular microservices, enabling flexible deployment and resource-efficient operation in IoT scenarios. The proposed contributions can be summarized as follows:

   - Development of a modular architecture where the LoRaWAN node functionalities (application layer, LoRaWAN MAC layer, and physical LoRa modulation) are each deployed inside independent Linux containers, managed through Docker and Docker Compose tools.
   - Integration of a MATLAB-based LoRa transmission model for the physical layer implementation, allowing a software-based realization of the modulation/demodulation processes typically handled by dedicated hardware.
   - Realization of a full prototype setup that couples the containerized architecture with a low-cost Software Defined Radio (SDR) device, highlighting the practical considerations, such as the throughput limitation between the host and SDR (40 MB/s).

This work's results show that while a fully software-defined and virtualized LoRaWAN node is technically feasible, the performance is influenced by the computational burden of the software-defined physical layer and the data transfer constraints between the SDR and the host system. Nonetheless, the modular design opens significant opportunities for flexible node deployment and dynamic reconfiguration in future LPWAN-based IoT networks.

3. **Performance Evaluation of Container-Based LoRaWAN Virtualization (Paper III):**

   To answer *Research Question 3*, we investigated whether container-based virtualization could be effectively used to virtualize LoRaWAN end nodes while maintaining performance and scalability in IoT deployments. The purpose is to assess if breaking down LoRaWAN node functionalities into modular containers could achieve similar operational efficiency to traditional monolithic or hardware-based nodes, while offering the added advantages of flexibility and scalability. The proposed contributions can be summarized as follows:

   - Setup of a controlled experimental testbed to measure resource usage, latency, and container behavior under varying computational resources.
   - Comparative analysis of MATLAB- vs. C-based implementations of LoRa baseband processing within containers.
   - Assessment of system-wide metrics including CPU and memory usage, process distribution, and container intercommunication overhead.
   - Contribution to the understanding of trade-offs in containerized wireless node virtualization, with relevance to scalable IoT system design.

   This work's results show that container-based virtualization is a promising approach for IoT end nodes, offering significant improvements in flexibility and deployment scalability. However, careful resource management and hardware abstraction techniques are critical to ensure that the performance requirements, especially for latency-sensitive applications, are fully met.

Collectively, these contributions deliver both theoretical advancements and practical implementations that address critical challenges in next-generation communication systems. By enhancing vehicular networking reliability and proposing a novel containerized IoT architecture, this thesis promotes more adaptive, scalable, and resource-efficient infrastructures. The findings presented herein lay a strong foundation for the development of robust and flexible wireless systems, facilitating future innovations in smart transportation and the IoT.

1.5 Thesis Structure

The remainder of the thesis is organized as follows:

- **Chapter 2** provides a comprehensive Background and Related Work, divided into two thematic parts. The first part covers C-V2X communications, with a focus on the limitations of the standard SB-SPS resource allocation method in out-of-coverage (OoC) scenarios, particularly for vehicle platooning applications. The second part introduces virtualization in LoRaWAN networks, reviewing the challenges of traditional hardware-dependent node implementations and the emergence of software-defined and container-based architectures.
- **Chapter 3** addresses the improvement of C-V2X communication reliability for vehicle platooning by introducing a novel dual-strategy approach. Related research results are described, which are published in Paper I.
- **Chapter 4** addresses the virtualization of LoRaWAN end nodes using container-based lightweight virtualization and SDR. The first phase 4.1 discusses the design and deployment of a fully software-defined LoRaWAN node architecture, and related research results are described, which are published in Paper II. The second phase 4.2 extends this work by analyzing CPU, memory, and timing metrics across different configurations (MATLAB-based and C-based LoRa implementations), with related results described in Paper III.
- **Chapter 5** concludes the thesis by summarizing the key findings, discussing the broader implications of the research outcomes, and proposing directions for future work that could further enhance vehicular and IoT communication systems.
- **Appendixes A,B, and C** include the first pages of the peer-reviewed papers published during the PhD work. These are serve as documented evidence of the research outcomes and contributions presented throughout the thesis.

1.6 List of Publications

This section presents the peer-reviewed publications that have resulted from the research conducted during my doctoral study. They collectively represent the principal findings, theoretical contributions, and methodological advancements developed throughout this research endeavor. Listed below are the peer-reviewed publications associated with this thesis. The full texts of these papers are also attached in the Appendixes A,B, and C section for further reference.

Peer Reviewed:

I **Khalilnasl, H.**; Iacono, S.D.; Ferrari, P.; Flammini, A.; McCarthy, B.; Sisinni, E. *Enhancing C-V2X Vehicle Platooning: A Dual-Strategy of Time Synchronisation and Partition-Based Scheduling*.. In Proceedings of the 2024 IEEE International Symposium on Measurements & Networking (M&N), 2024, pp. 1-6. https://doi.org/10.1109/MN60932.2024.10615853.[1]

II **Khalilnasl, H.**; Depari, A.; Ferrari, P.; Flammini, A.; Gaffurini, M.; Sisinni, E. *Implementing a Software Defined LoRaWAN Node Exploiting Container-Based Lightweight Virtualization*. In Proceedings of the 2024 IEEE International Symposium on Measurements & Networking (M&N). IEEE, 2024, pp. 1-6. https://doi.org/10.1109/MN60932.2024.10615549. [2]

III **Khalilnasl, H.**; Ferrari, P.; Flammini, A.; Sisinni, E. *On the Use of Containers for LoRaWAN Node Virtualization: Practice and Performance Evaluation* . In Electronics 14, no. 8: 1568. https://doi.org/10.3390/electronics14081568.[3]

Chapter 2

Background and Related Work

2.1 Introduction

This chapter provides an overview of the key concepts, background, and related work relevant to the topics addressed in this thesis. The section is divided into the following subsections:

In Section 2.2, we provide an Overview of Wireless WAN Technology, where we introduce the fundamental principles and evolution of wireless communication technologies, laying the groundwork for understanding their role in modern networked systems. In Section 2.3, we present the Background: C-V2X, focusing on the key concepts of V2X technology, particularly its application in vehicle platooning and the communication challenges associated with OoC scenarios. In Section 2.4, we provide a Literature Review: C-V2X, discussing recent advancements in C-V2X communication, examining the existing challenges, and analyzing the state-of-the-art solutions proposed to address those challenges, particularly in vehicle platooning and safety-critical applications. Following this, in Section 2.5, we offer the Conclusion: C-V2X, summarizing the key findings and insights on the use of C-V2X for vehicular networks and the proposed solutions that enhance its performance.

In Section 2.6, we present the Background: LoRaWAN, detailing the LoRaWAN technology, its architecture, and its relevance for IoT applications. We emphasize the challenges related to the traditional, hardware-dependent implementation of LoRaWAN nodes and how these limitations affect scalability and flexibility. In Section 2.7, we explore the Literature Review: LoRaWAN, where we examine the state of the art in LPWAN technologies, with a specific focus on LoRaWAN. We discuss the challenges and innovations in the field, particularly related to the lightweight virtualization of LoRaWAN nodes, and how containerization offers a promising solution to enhance flexibility and resource management. Finally, in Section 2.8, we offer the Conclusion: LoRaWAN, summarizing the research contributions to the LoRaWAN field, particularly regarding the virtualization of LoRaWAN nodes, and proposing directions for future work to overcome the challenges identified in the current systems.

2.2 Overview of Wireless WAN Technology

Wireless Wide Area Networks (WWANs) have become integral to the development of modern communication systems, enabling seamless connectivity over large geographical areas. These networks, which span cities, countries, and even continents, provide the backbone for a wide array of applications ranging from mobile broadband to machine-to-machine (M2M) communication. Over the years, advancements in wireless technologies have made it possible to connect billions of devices in various sectors, including transportation, agriculture, smart cities, and industrial applications. At the core of WWANs technologies are the principles of wireless communication, which allow data to be transmitted over long distances without the need for physical connections.

At the highest level, Cellular communication, specially C-V2X and LoRaWAN can be conceptualized as domain–specific instantiations of WWANs–a category of communication technologies characterized by their ability to provide wireless connectivity over large geographic areas. As modern WWANs evolve beyond their traditional role in mobile broadband service, they are increasingly optimized for vertical-specific use cases, particularly in the context of autonomous systems, smart infrastructure, and the IoT. Within this emerging landscape, C-V2X and LoRaWAN represent two diverging, yet complementary, trajectories in the design space defined by the latency-energy-range trade-off.

On one end of this spectrum, **C-V2X**–standardized by 3rd Generation Partnership Project (3GPP) as part of the broader 4G/5G evolution–prioritizes ultra-reliable, low-latency communication (URLLC) to support the safety-critical requirements of vehicular networks. It leverages cellular infrastructure and dedicated sidelink communication (PC5 interface) to enable real-time coordination among vehicles, Roadside Unit (RSU)s, and cloud services. As such, C-V2X extends the WWAN model toward high-mobility, latency-sensitive environments, where millisecond-scale responsiveness and deterministic performance are paramount.

At the opposite end lies **LoRaWAN**, an LPWAN protocol developed under the LoRa Alliance, which is optimized for low-power, long-range, and low-throughput communications. LoRaWAN caters to massive-scale deployments of battery-powered devices, often in static or low-mobility environments such as smart agriculture, environmental monitoring, and utility metering. In contrast to C-V2X, it sacrifices data rate and latency to achieve extended device lifetime and cost-effective scalability across thousands or millions of end nodes.

Thus, while C-V2X and LoRaWAN serve fundamentally different application domains, they both illustrate how WWAN technologies are diversifying to accommodate the heterogeneous requirements of next-generation networks. Their coexistence reflects a paradigm shift from a one-size-fits-all communication model to a heterogeneous and layered architecture, where multiple WWAN technologies operate in tandem to deliver tailored connectivity across an expanding array of verticals.

This section provides an overview of WWAN technologies, focusing on the relative stage of maturity of key standards, the positioning of C-V2X and LoRaWAN within the broader WWAN landscape, and the inherent trade-offs between different wireless communication technologies. By exploring these dimensions, the chapter lays the conceptual foundation for the performance enhancement strategies proposed later in this thesis. It aims to demonstrate not only the technical distinctiveness of C-V2X and LoRaWAN, but also their shared role in advancing the vision of ubiquitous, intelligent connectivity.

WWANs are essential for providing connectivity over large distances, allowing devices to communicate without physical infrastructure. These networks support a wide range of applications, from cellular communications to the IoT, enabling systems to operate in cities, regions, or even entire countries. A key characteristic of WWANs is their ability to cover large geographical areas, often over 100 km, depending on the specific technology. The landscape of WWANs technologies is di-

Table 2.1: Positioning of C-V2X and LoRaWAN within the WWAN Landscape

Aspect	**C-V2X**	**LoRaWAN**
WWAN Subclass	Cellular WWAN (3GPP)	LPWAN under WWAN (LoRa Alliance)
Spectrum Type	Licensed (e.g., 5.9 GHz ITS band)	Unlicensed ISM bands (e.g., 868 MHz, 915 MHz)
Primary Application Domain	Vehicular communication	Massive IoT deployments
Mobility Support	High (supports high-speed, mobile nodes)	Low (best suited for static or low-mobility devices)
Communication Model	Both network-assisted and direct (sidelink PC5)	Asynchronous, star topology
Power Constraints	Moderate to high (vehicle-powered devices)	Ultra-low (battery-powered devices)
Data Rate	High (up to 100 Mbps with 5G enhancements)	Very low (∼0.3–50 kbps)
Latency Requirements	Ultra-low (ms-level, safety-critical)	High latency tolerated (seconds)
Integration Ecosystem	Part of 4G/5G and ITS infrastructure	Independent IoT networks, often with public gateways
Deployment Maturity	Emerging / under trial in several countries	Mature / globally deployed

verse, with various solutions designed to meet specific needs such as long-range communication, low power consumption, and high data rates.

To better illustrate the complementary nature of C-V2X and LoRaWAN within the evolving WWAN landscape, a comparative summary is provided in Table 2.1. This contrast exemplifies the broader trend toward heterogeneous WWAN architectures designed to serve diverse application domains.

Fig.2.1 illustrates the various technologies related to LPWAN. It categorizes these technologies into two main groups: Proprietary and Standards. Each of these categories includes various sub-technologies, standards, and protocols that are relevant for deploying LPWAN-based applications, particularly in the IoT domain.

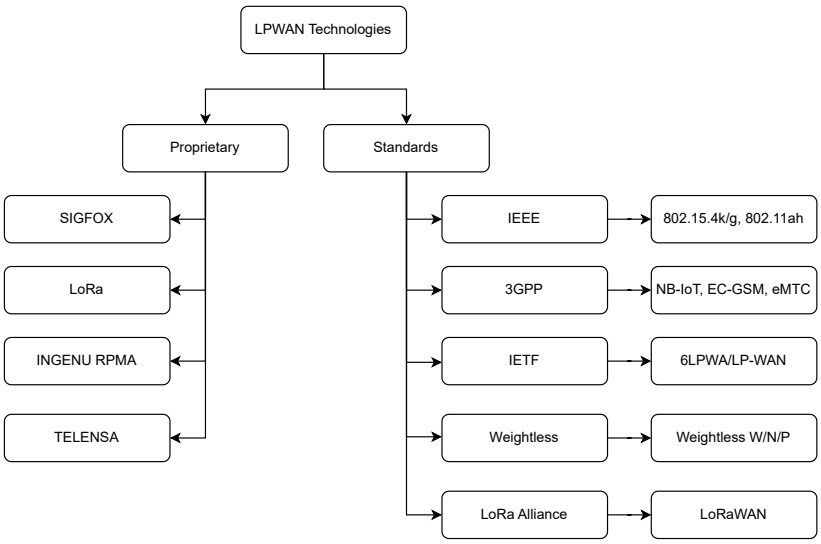


Figure 2.1: Overview of LPWAN technologies

- **Proprietary LPWAN Technologies:** Proprietary technologies are those that are developed and maintained by individual companies or organizations. These are often designed to provide specific solutions with a focus on performance,

reliability, or cost-effectiveness for specific applications. In the Fig.2.1, we see the following proprietary LPWAN technologies:

  - *SIGFOX*: It operates on unlicensed sub-GHz bands, using ultra-narrowband (UNB) modulation to optimize spectrum use and minimize energy consumption. The technology is primarily designed for low-data-rate communications and can support a large number of devices over wide areas. SIGFOX's simplified communication approach reduces operational costs, though its message size and frequency are constrained by regional regulations [4, 5, 6, 7].
  - *LoRa*: LoRa (Long Range) is one of the most popular proprietary technologies in the LPWAN category. Developed by Semtech, it uses a spread-spectrum technology called Chirp Spread Spectrum (CSS) for long-range, low-power communication, ideal for IoT applications where minimal data needs to be transmitted over large distances [8]. We will explain more about this in Chapters 2 and 4.
  - *INGENU RPMA (Random Phase Multiple Access)*: This technology is designed for secure and reliable LPWAN communication in remote areas. It operates on a narrowband frequency and ensures robustness against interference, making it ideal for large-scale sensor networks in challenging environments [9].
  - *TELENSA*: A proprietary LPWAN technology designed for smart meter and other IoT applications. TELENSA focuses on providing efficient, long-range communications with minimal power usage, especially for urban and rural utility applications [10, 6].

- **Standard-Based LPWAN Technologies:** In contrast to proprietary technologies, standard-based LPWAN solutions are defined by international standards organizations. These technologies are developed to ensure compatibility, interoperability, and open standards, which allow different devices from various manufacturers to communicate with each other over the same network. The chart highlights the following standard-based technologies:

  - *Institute of Electrical and Electronics Engineers (IEEE)*: IEEE's LPWAN protocols include 802.15.4k, which is used for low-rate wireless personal area networks (LR-WPANs), and 802.11ah, a Wi-Fi standard optimized for low-power, wide-area communication.
  - *3GPP*: The 3GPP is a key player in defining cellular LPWAN standards. Notably, NB-IoT (Narrowband IoT), EC-GSM (Extended Coverage GSM), and eMTC (enhanced Machine Type Communications) are the major LPWAN technologies defined by 3GPP, focusing on enhancing cellular networks to support IoT devices with low power and wide-area coverage [11, 12].
  - *IETF (Internet Engineering Task Force)*: IETF standards focus on the Internet Protocol (IP) suite for LPWANs, enabling the interoperability of LPWAN technologies with Internet-based protocols. This includes the

6LPWA (IPv6 Low Power Wide Area Networks) standard, which facilitates the use of IPv6 addresses in LPWANs.

- LoRa Alliance: While LoRa is a proprietary technology, its application is often standardized under the LoRaWAN protocol, which is an open, widely adopted protocol supported by the LoRa Alliance. LoRaWAN ensures interoperability and open access, providing a well-defined network architecture for building LPWANs based on LoRa technology [13, 14].
- *Weightless*: This is an open standard for LPWANs, supporting multiple variants, such as Weightless W, N, and P. It is designed for long-range, low-power communication and provides flexibility for IoT applications, including smart cities and agriculture.

2.3 Background: C-V2X

2.3.1 Introduction

This section is structured into two distinct sections. The first section provides a comprehensive overview of the technical background of the vehicular communication technologies explored in this thesis, laying the foundation for understanding their relevance and application in modern transportation systems. It covers key concepts, the evolution of communication standards, and the role these technologies play in enabling innovations such as vehicle platooning and autonomous driving. The second section offers a thorough review of the current state-of-the-art literature, focusing on the research challenges and advancements in the field. This includes a detailed analysis of resource allocation techniques, scheduling algorithms, and the performance evaluation of C-V2X and other related vehicular communication systems.

2.3.2 Vehicular Networking

Vehicular networks refer to the integration of communication technologies within the context of vehicles. The primary aim is to enable vehicles to interact with other vehicles, roadside infrastructure, and even vulnerable road users, like pedestrians. This communication system is commonly known as Vehicle-to-Everything (V2X). The primary focus of vehicular networks is enhancing safety by minimizing the risk of accidents, followed by improving traffic flow and, ultimately, providing infotainment services. Safety-related applications often rely on Vehicle-to-Vehicle (V2V) or Vehicle-to-Pedestrian (V2P) communications. Traffic management may involve V2V but also includes Vehicle-to-Infrastructure (V2I) communication. For infotainment services, such as video streaming, Vehicle-to-Network (V2N) communication is essential. These communication types are illustrated in Fig. 2.2.

The initial phase of deploying these services is referred to as ”Day 1” services, which are currently being rolled out in pilot projects like C-Roads in the European Union [15]. The focus of Day 1 services is primarily on the two main areas: safety and traffic efficiency. A fundamental component of Day 1 services is the Cooperative Awareness (CA) service, responsible for sending Cooperative Awareness Messages (CAMs). This application is essential for safety, as it communicates the characteristics and movements of vehicles in relation to nearby vehicles to prevent collisions. As deployments progress and penetration rates rise, this service will become a key part of Advanced Driver Assistance Systems (ADAS). While the CA service is the primary focus of this thesis, the Dynamic Event Notification (DEN) service also plays a critical role in safety, particularly in emergency scenarios like sudden braking or the approach of emergency vehicles.

The first primary service, CA, forms the backbone of traffic in any V2X network for all ”Day 1” deployments and is standardized in [16]. CA focuses on vehicles sharing their kinematic data with neighboring vehicles to help prevent collisions.

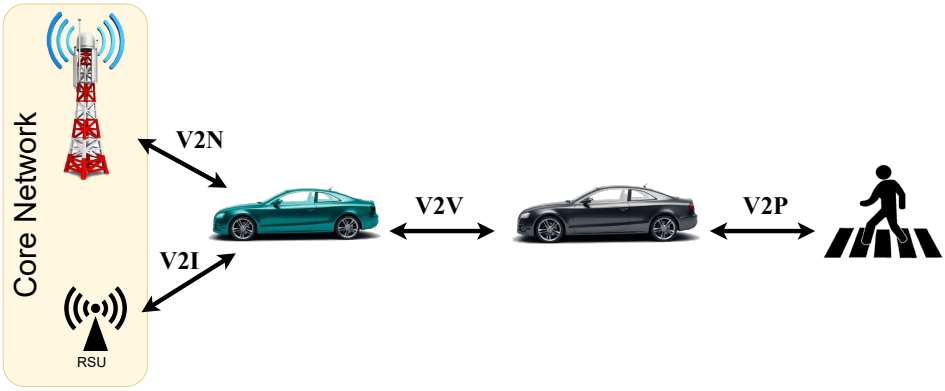


Figure 2.2: V2X communication patterns

Accordingly, CAMs include information such as the vehicle's position, speed, heading, and dimensions. Another service examined in this study is DEN, which encompasses a wider range of services with varying requirements [17]. Generally, DEN services are intended for irregular or emergency communications, with the most common examples being vehicle breakdowns or the approach of an emergency vehicle. Additionally, DEN services are used for other diverse applications, such as weather advisories or roadwork warnings, with the C-Roads platform specifically designed for such applications.

The first vehicular networking technology introduced was based on the 802.11p wireless protocol. This eventually led to the development of the ITS-G5 standard in Europe, proposed by ETSI [18], and Wireless Access in Vehicular Environment (WAVE) in North America [19]. Although this technology is well-understood and deeply embedded in the standardization process, its deployment has been slower than anticipated. As a result, 3GPP introduced a competing standard in LTE-V2X, also known as C-V2X, as part of 3GPP release 14 [20]. This was the initial version, which has since evolved into the NR-V2X standard, based on C-V2X [21].

Backed by the efforts of the 5G Automotive Association (5GAA), the 3GPP standard has witnessed growing support as it effectively addresses the persistent issue of channel congestion that has limited the adoption of 802.11p-based systems. Notably, Release 14 introduced a novel sidelink interface, designated as PC5, to facilitate direct vehicle-to-vehicle V2V communication. This development extended from the LTE Proximity Services (ProSe) defined in Release 12 for Device-to-Device (D2D) connectivity [22]. The release also delineated two operational communication modes-Mode 3 and Mode 4-as well as introduced new architectural components. As illustrated in Fig. 2.3, Mode 3 operates under centralized control, where the cellular infrastructure is responsible for resource allocation and scheduling for PC5-based V2V transmissions through the Uu interface. In contrast, Mode 4 allows for autonomous operation, where vehicles independently choose their radio

resources by employing a decentralized scheduling mechanism known as SB-SPS. This mode eliminates dependence on network infrastructure, mirroring the functionality of wireless ad hoc networks. Mode 4 is particularly critical for safety-related applications, which must function even in the absence of cellular coverage, thereby positioning it as a viable alternative to DSRC/WAVE and ITS-G5 systems. This study will specifically focus on Mode 4's autonomous resource selection, as well as the subsequent NR-V2X Mode 2 standard, which builds on it.

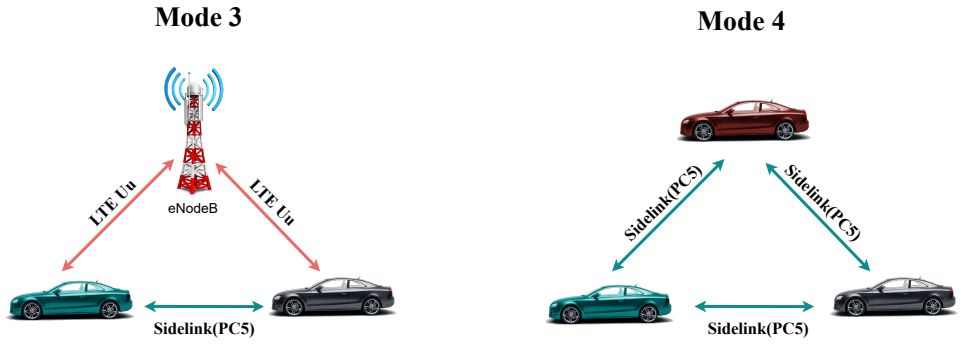


Figure 2.3: Resource Allocation Modes in LTE-V2X

2.3.3 Spectrum Allocation

One of the primary concerns and significant regulatory challenges with any wireless technology is the allocation of spectrum for specific tasks. The spectrum allocation for Intelligent Transport Systems (ITS) varies globally, but the crucial point is that ITS-G5, C-V2X, and NR-V2X all operate within the sub-6 GHz spectrum range. Initially, spectrum allocations would have favored ITS-G5 as the preferred technology, which is why the spectrum ranges are often allocated to different tasks or functional areas in various regions. ITS-G5 typically provides multiple 10 MHz channels, each of which can be dedicated to specific tasks or application types. Vehicles can choose to listen to certain sections of the spectrum if they are interested in a particular application. In contrast, C-V2X and NR-V2X operate differently, with channel bandwidths ranging from 10 to 20 MHz for a specific sidelink configuration. Ultimately, the future of these technologies depends on regulatory decisions-will they coexist, or will one dominate and have exclusive access to the spectrum? These questions are beyond the scope of this study, but it is important to note that the spectrum is limited, posing challenges for any technology that is adopted. The spectrum allocations in regions like the EU, USA, China, and Japan. In the EU, 70 MHz of spectrum is dedicated to ITS applications, divided into four categories of use. Specifically, for safety-related applications, the EU allocates 20 MHz of spectrum. This is the maximum bandwidth allowed for C-V2X to use in a single contiguous segment for

sidelink communication, and it plays a crucial role in managing congestion. While 20 MHz is a significant amount, it is not infinite. As this study will demonstrate, even with 10 MHz, a typical application running at a 10 Hz rate in a congested scenario can result in substantial channel congestion. Considering the various safety applications-such as CA, Cooperative Perception (CP), and DEN-and any additional services added on top, it is vital to account for the diversity of vehicular scenarios, particularly in high-density traffic conditions.

2.3.4 Scheduling Feature in C-V2X

C-V2X Mode 4, as defined by 3GPP, utilizes the SB-SPS algorithm for resource allocation at the MAC and PHY layers which is more detailed in [23, 24, 25, 26]. This process can be divided into three key phases: Selection and Sensing, and Transmission. However, before delving into these phases, it is essential to first understand the channel structure and the information used in the scheduling process.

- **Channel Structure:** The C-V2X channel is based on Single-Carrier Frequency Division Multiple Access (SC-FDMA). In the **time domain**, resources are organized into subframes of 1 ms, which are grouped into 10 ms frames. Each subframe consists of 14 SC-FDMA symbols: 4 symbols are dedicated to demodulation reference signals, 1 symbol for Tx-Rx switching, and the remaining 9 symbols are allocated for data transmission.

  In the **frequency domain**, the channel is divided into 15 kHz subcarriers, which are grouped into Resource Blocks (RBs). Each RB contains 12 subcarriers and spans 1 subframe. Unlike conventional LTE resource structures, C-V2X groups RBs into subchannels. The number of RBs per subchannel and the total number of subchannels are configurable, but they are constrained by the allocated bandwidth, which can be 10 or 20 MHz.

  This structure is illustrated in Fig. 2.4.

  The channel is also divided into the Physical Sidelink Control Channel (PSCCH) and the Physical Sidelink Shared Channel (PSSCH), both consisting of RBs. The PSSCH is responsible for transmitting application layer/data packets, referred to as Transport Blocks (TBs), while the PSCCH transmits scheduling control information known as Sidelink Control Information (SCI).

  These channels can operate in either non-adjacent or adjacent modes. In adjacent mode, as shown in Fig. 2.5a, the PSCCH is incorporated within each subchannel alongside the PSSCH. This configuration is advantageous because it requires only 2 RBs for SCI transmission when multiple subchannels are assigned, allowing the remaining PSCCH RBs to be allocated to the PSSCH. In the non-adjacent mode, as shown in Fig. 2.5b, a dedicated segment of the channel is allocated to the PSCCH, with the remaining portion used for the PSSCH. In this case, the PSCCH section cannot be reassigned to the PSSCH when multiple subchannels are allocated. Throughout this study, the adjacent mode configuration is used.

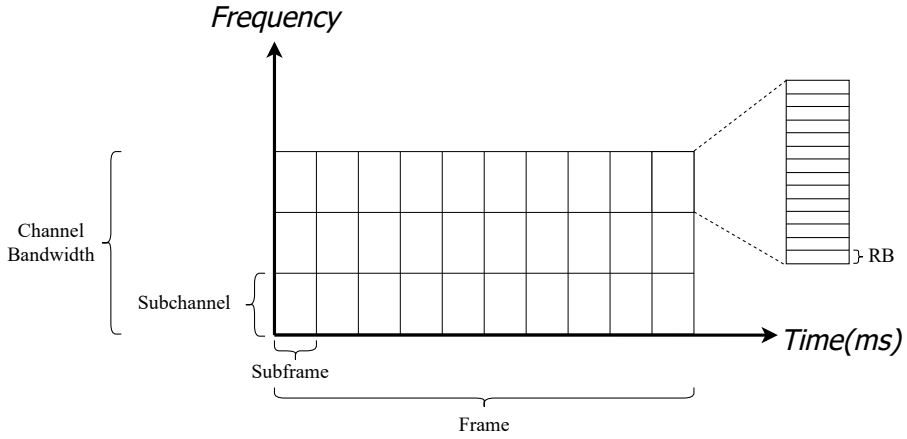


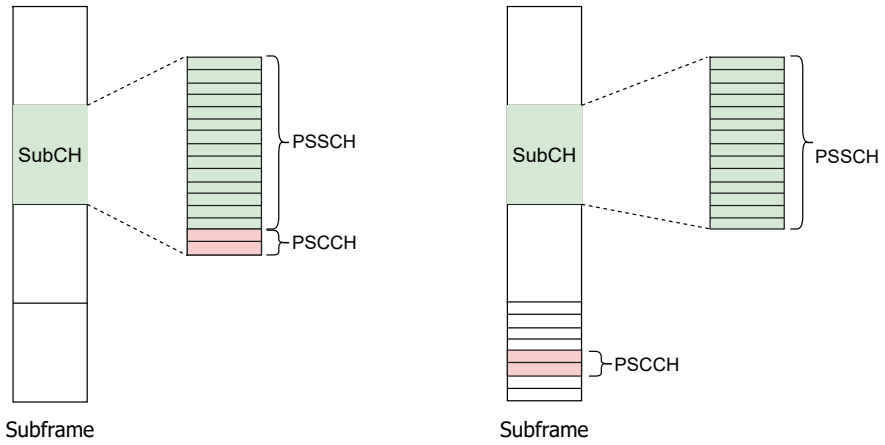
Figure 2.4: C-V2X Channel Structure

The structure of SCI Format 1 is shown in Table 2.2. SCI Format 1 provides some details such as packet priority (Priority field), the resource reservation time (Resource Reservation field), the number and location of resources for both initial transmission and retransmission, retransmission timing (Time gap), Modulation and Coding Scheme (MCS), and the retransmission status (Retransmission index field). An SCI is transmitted within the same time slot as a Transport Block (TB), occupying 2 RBs. If the SCI is not received, the associated TB cannot be decoded, as the MCS is specified in the SCI.

Table 2.2: 3GPP SCI Fields

Priority	Resource Reservation	Frequency Resource location of initial and retransmission	Time Gap	Modulation and coding scheme	Retransmission index
3 bits	4 bits	Calculation based on the number of subchannels	4 bits	5 bits	1 bit

- **Sensing Window & Selection Window:** Within the MAC layer of C-V2X, autonomous resource selection is facilitated through the SB-SPS mechanism, identified as Mode 4. This procedure is initiated when data arrives from higher layers. In response, the MAC layer formulates a scheduling grant that outlines both the number of subchannels required and the interval for repeated transmissions. If a grant has previously been established, incoming data packets are assigned according to this existing schedule. The determination of the number of subchannels is influenced by factors such as the current network configuration and specific application demands, including the size of the data packets and the MCS in use. The repetition interval, governed by the Resource Reservation Interval (RRI), is defined by upper-layer parameters. While RRIs



a) Adjacent b) Non-adjacent

Figure 2.5: Adjacency modes

typically range from 100 to 1000 milliseconds in 100-millisecond steps, certain V2P scenarios permit shorter intervals of 25 or 50 milliseconds [27]. Following the generation of the grant, the PHY layer is tasked with identifying a set of subchannels that conform to the grant's specifications. These candidate selections, termed Candidate Single-Subframe Resources (CSRs), can include multiple subchannels grouped within a single subframe. The selection process considers all possible CSRs that fall within a predefined Selection Window, which begins when the data packet is received and extends up to the latency threshold prescribed by upper-layer configurations.

The first stage of Candidate Single-Subframe Resource (CSR) filtering addresses the Half-Duplex (HD) challenge for C-V2X. Since the antennas used in C-V2X and NR-V2X cannot simultaneously transmit and receive, the vehicle cannot detect collisions or check if other subchannels are occupied during transmission. Therefore, the SB-SPS algorithm discards any subframe in the Selection Window that overlaps with the subframe in which the vehicle transmitted, based on the configured RRI.

Subsequently, the initial list of CSRs undergoes refinement based on SCI gathered during the Sensing Window, which encompasses the most recent 1000 subframes. Resources are excluded from consideration if the SCI suggests that a given CSR will be occupied during the upcoming Selection Window. Additionally, CSRs are removed if their corresponding PSSCH Reference Signal Received Powe (RSRP) surpasses a specified threshold. The RSRP value represents the signal strength of a received transmission. After these filtering

steps, it is required that no fewer than 20% of the original CSRs remain eligible. Should the available pool fall short of this requirement, the threshold for RSRP is incrementally increased by 3 dB, and the filtering procedure is repeated. In the final selection stage, the Physical layer identifies and selects the 20% of CSRs with the lowest average Sidelink Reference Signal Strength Indicator (RSSI), as measured across the entire duration of the Sensing Window. RSSI indicates the total noise on a subchannel, including RSRP, interference, and background noise, ensuring that the CSRs with the least interference are selected.

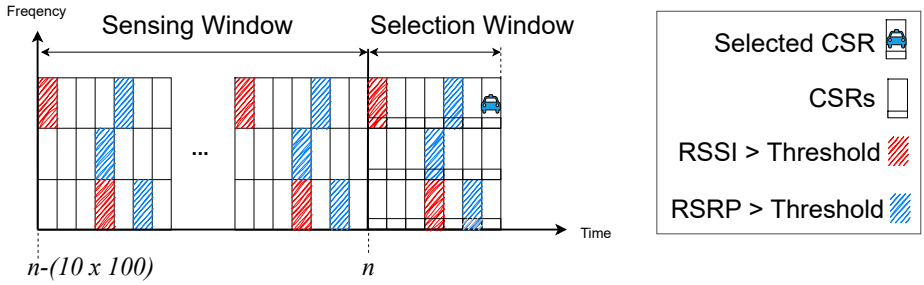


Figure 2.6: SB-SPS mechanism

The grant process is illustrated in Fig. 2.6, providing a concrete example of its operation. At time n, the MAC layer requests a set of CSRs from the PHY layer. Assuming this is the first transmission, the PHY layer generates a Selection Window with no resources discarded due to HD. Next, the SCI and RSRP filtering stage occurs, where resources (blue) are removed because their average PSSCH RSRP exceeds the threshold. The SB-SPS mechanism then progresses to the RSSI filtering stage, where resources (red) are filtered out due to high recorded RSSI on these subchannels. Consequently, the MAC layer is left with the remaining resources to select from. The filtered set of CSRs is then forwarded to the MAC layer, where a single CSR is selected at random. This randomness helps reduce the chances of resource collisions between vehicles. The chosen CSR is allocated for a series of repeated transmissions, the count of which is governed by the Resource Reselection Counter (RC). This counter is initialized with a randomly selected integer from the range 5, ..., 15, and it decreases by one after each subsequent transmission. Once the RC reaches zero, the vehicle has two possible actions: with probability P, it may continue to use the same CSR; otherwise, it triggers a fresh resource selection by initiating a new scheduling grant, thereby repeating the entire selection procedure. The value of P, previously configured, can range from 0 to 0.8. This process is depicted in Fig. 2.7, where the blue car has selected a CSR. A key parameter in this process is the Resource RRI, which governs the time period between transmissions and enables the SB-SPS mechanism to maintain an accurate history of resource selection by neighboring vehicles. It is important to note that the RC

can also take values between {25, ..., 75} and {15, ..., 30} for RRI below the 100 ms interval, though these cases are not considered in this study.

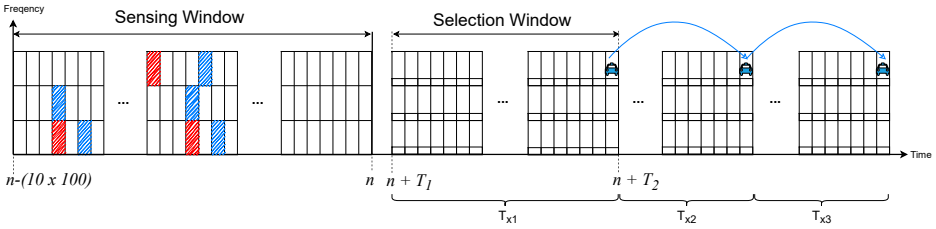


Figure 2.7: Grant mechanism

When a TB is ready for transmission, the MAC layer must select a suitable MCS from the predefined range. If none of the available MCS options can satisfy the requirements for the allocated resource-particularly if the highest MCS still falls short-the resource is released, and the selection process recommences to identify a new one. However, if a compatible MCS is found, the MAC layer forwards the TB to the PHY layer, accompanied by the information required to construct the corresponding SCI message.

Both the TB and SCI are then transmitted within the same subframe by the PHY layer. Meanwhile, the MAC layer reduces the reservation counter, which tracks how many transmissions remain for the current resource. When this counter reaches a value of 1, the MAC layer evaluates whether to retain the resource. This decision is based on a probabilistic check: the resource is retained only if a randomly drawn value X exceeds the threshold $probResourceKeep$, where $probResourceKeep$ lies within the interval $[0, 0.8]$. If the condition is not met, the resource is released. In this case, the SCI message accompanying the final TB transmission explicitly indicates that the resource has been vacated by setting the RRI field to zero.

The SCI is a crucial component of the SB-SPS mechanism, as detailed in Table 2.2. It specifies the resources or subchannels occupied by the TB and, since it includes the MCS, the TB cannot be decoded without first decoding the SCI. One of the primary fields of interest in this study is the RRI field, which controls the time interval between transmissions. The RRI is essential for the PHY layer to track which resources in the Selection Window are reserved by neighboring vehicles.

2.3.5 Key changes from LTE-V2X to NR-V2X

In general, NR-V2X Mode 2, as introduced in 3GPP Release 16, adopts a methodology comparable to that of Mode 4 but incorporates several enhancements at the physical layer. One key improvement is the adoption of flexible numerology, which allows for adjustable subcarrier spacing-an important feature for achieving reduced

transmission latency. Furthermore, the standard introduces new communication channels tailored for both groupcast and unicast transmissions. To support these enhanced communication modes, a revised multi-stage SCI structure has been developed. This adaptation facilitates more granular control over resource allocation and signaling. Readers interested in a more comprehensive overview of the PHY-layer advancements in NR-V2X are directed to [28].

At the MAC layer, several important modifications have been introduced that influence the performance of SB-SPS. One of the key adjustments pertains to how RSRP is processed. Unlike Mode 4, which calculates an average RSRP value based on the entire sensing window, Mode 2 bases its decision on the RSRP from only the most recent transmission when identifying reserved resources.

Another notable change is the elimination of the RSSI filtering stage. As a result, every CSR that remains after the RSRP-based filtering is forwarded to the MAC layer for possible selection. These alterations have a substantial impact on how effectively SB-SPS handles irregular or event-driven traffic patterns, a topic explored in detail in [29].

NR-V2X introduces additional mechanisms not directly explored in this study, specifically the **re-evaluation** and resource **pre-emption** mechanisms.

Re-evaluation allows a vehicle to defer the use of a selected CSR, even after it has been chosen, if it detects another vehicle that has reserved the resource and its use would result in a collision. This mechanism is particularly important in NR-V2X, as the RRI mechanism has been adapted to support sub-100 ms RRIs, configurable between 1 and 50 ms, unlike C-V2X, which only supports fixed RRIs of 100, 200, or 300 ms.

The pre-emption mechanism enables a vehicle to defer using a selected CSR if it senses that a neighboring vehicle, operating a higher priority service, requires the resource. It is important to note that this study has a limited focus on packet prioritization.

2.3.6 Platooning Vehicles and groupcast in C-V2X

A platoon refers to a group of vehicles that move in a coordinated manner, typically led by a lead vehicle as shown in Fig. 2.8. The vehicles in the platoon maintain short distances between each other to improve road safety and increase highway capacity. The platoon structure often follows a leader-follower model where the lead vehicle dictates the speed and maneuvering, while the trailing vehicles adjust their actions accordingly to maintain synchronization fits are wide-ranging, such as enhancing fuel efficiency, road safety, and traffic flow. Vehicle platooning is not a new concept; it has been around for several decades. General Motors first introduced the idea in 1939 at the New York World's Fair with their film *To New Horizons* [30]. The primary benefits of platooning include fuel conservation, energy efficiency, and pollution reduction [31]. The concept envisions self-driving vehicles using automatic radio control to maintain safe distances and travel at a constant speed, while curved sides help them stay in their lanes. Early research on vehicle platooning focused more on

traffic dynamics than on autonomous vehicle control. The next phase of the research involved analyzing how different spacing policies in heavy-duty vehicle platoons affect traffic flow. Numerous studies have since been conducted on automobile platooning, examining varying penetration rates of vehicles equipped with platooning technology, as well as its impact on traffic flow, where the initial platooning studies, particularly on string stability, originated.

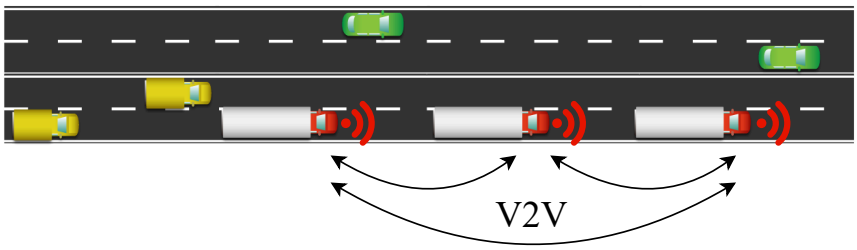


Figure 2.8: Traffic scenario including a platoon with V2V communication approach

When vehicles are grouped closely together in a platoon, vehicle platooning is expected to enhance fuel efficiency [32, 33, 34] and alleviate traffic congestion [35, 36, 37] by reducing air resistance for platoon members. Experimental studies show that a bus following another vehicle in a specific convoy arrangement can reduce aerodynamic drag by up to 40% at 80 km/h, provided the gap between vehicles is maintained at 10 meters [38]. As a result, it is reasonable to assume that companies providing chipsets, communication systems, mobile operators, infrastructure suppliers, and automakers are actively seeking ways to improve mobility and transportation services. In this context, platooning systems present a practical strategy for deploying fuel-efficient alternatives. Beyond safety benefits, platooning has the potential to reduce traffic, improve fuel efficiency, and enable faster, more effective travel [32]. These systems can decrease fuel consumption by as much as 10% [31], depending on factors such as travel distance, platoon speed, and size. The primary benefits come from reducing air drag resistance and coordinating the acceleration and deceleration of platoon members.

Several key factors enable the benefits of platooning systems. First, increasing automation driven by advancements in computer systems, sensors such as Global Positioning System (GPS) units, radars, cameras, and LIght Detection And Ranging (LIDAR), which monitor the surroundings and make critical decisions, play a central role. Second, vehicular communication networks are essential to achieve the high level of cooperation and coordination needed for such automation. Without these networks, platooning systems would not be feasible. The main challenge of sensor-based systems-Line-of-sight (LOS) issues-has been addressed by wireless communication technologies. Numerous studies on platooning have been conducted worldwide, including projects by California Partners of Advanced Transit and Highways (PATH) [39], CHAUFFEUR I [40], GCDC [41, 42, 43], and ENSEMBLE [44].

According to [45], minor adjustments to CAMs or DEN messages may be sufficient for the implementation of platooning. To achieve the close distances required between platoon members, the authors suggest removing the variable CAM rate and fixing it at the maximum value of 10 Hz. Therefore, platooning does not require significant changes to the existing message types. However, the timeliness of the received data is crucial for the successful and safe operation of platooning.

Cooperative Adaptive Cruise Control (CACC) is an advanced vehicular technology that enables automated and coordinated driving among a group of vehicles, often referred to as platooning. Unlike traditional Adaptive Cruise Control (ACC), CACC leverages V2V communication to share real-time speed and positional data, significantly enhancing traffic flow efficiency and reducing fuel consumption through tighter inter-vehicle spacing and more responsive control [46]. CACC has shown particular promise in truck platooning, where consistent speeds and minimal headways yield economic and environmental benefits [47]. Recent research highlights the development of eco-CACC strategies that optimize platoon behavior over varying traffic scenarios, further contributing to energy savings and emission reductions [48]. Moreover, challenges such as managing platoons on gradients or integrating human drivers into partially automated systems continue to drive innovations in controller design and optimization algorithms [49, 50]. These advancements underscore the potential of CACC as a cornerstone of future intelligent transportation systems. More recent references on CACC, particularly under realistic communication constraints and advanced control models, can be found in the works of [51, 52].

Studies have addressed platoon formation [53, 54], which also relies on SL. The vehicles communicate continuously with each other using wireless communication technologies, including V2V communication. The Day 1 Cooptelligent Transport Systems (C-ITS) technologies, such as CAMs, play a pivotal role in platooning operations. These technologies allow the platooning vehicles to communicate in real-time, ensuring that the necessary information for collision avoidance and synchronization is transmitted efficiently. Furthermore, other technologies like Dedicated Short Range Communications (DSRC) and the more recent Cellular Vehicle-to-Everything (C-V2X) are used for reliable communication across the platoon. The use of V2X communication, combined with CACC systems, helps ensure that each vehicle can react with minimal delay, significantly reducing the reaction time compared to human-controlled vehicles .

The primary technologies enabling platooning include CACC systems, where vehicles exchange data such as their position, speed, and planned maneuvers .

3GPP Release 16 is the first specification for NR-V2X, introducing three types of communication: unicast, broadcast, and groupcast (multicast). It also supports advanced use cases such as platooning, extended sensors, advanced driving, and remote driving. The work in [55] provides a summary of the key aspects of Release 16 and the relevant V2X features.

2.4 Literature Review: C-V2X

This section provides a comprehensive review of the current literature on C-V2X communications, with a particular focus on the key developments, challenges, and advancements identified in recent studies. The review is organized into several subsections, each addressing specific aspects of C-V2X technology and its application in vehicular networks.

First, we examine the evolution of C-V2X, emphasizing the progress from early LTE-V2X standards to the more recent NR-V2X protocols. This includes a detailed analysis of the key technological advancements that enable high-reliability, low-latency communications, which are essential for applications like vehicle platooning and autonomous driving.

Next, the review delves into the challenges associated with resource allocation and scheduling in C-V2X systems, particularly in the context of high-density traffic scenarios. A thorough exploration of Mode 4 communications and the role of scheduling algorithms such as SB-SPS in optimizing resource use is presented. We also discuss studies that address packet collisions, interference, and the overall performance of these scheduling techniques in ensuring efficient and reliable communication.

The section further investigates the integration of C-V2X with vehicle platooning, highlighting advancements in CACC and the role of groupcast communication in ensuring reliable intra-platoon communication. Specific research related to platooning communication performance, latency reduction, and the impacts of vehicular density on communication quality are explored.

Finally, a review of various simulation approaches and platforms used to evaluate C-V2X performance is included. This section highlights the methodologies employed by researchers to model and assess the effectiveness of C-V2X technologies, providing insights into how these models can be applied to future research and development in the field of vehicular communications.

2.4.1 Mode 4

Molina-Masegosa and Gozalvez [56] provide a comprehensive overview of the LTE-V2X communications, focusing on modes 3 and 4. The paper compares LTE-V2X with other technologies like IEEE 802.11p and discusses its role in supporting autonomous vehicles and connected transportation. One of the significant differences between Mode 3 and Mode 4 is how they handle resource allocation. The study further discusses congestion control in Mode 4, a critical feature to ensure that the system can handle high vehicle densities by managing the Channel Busy Ratio (CBR) and Channel occupancy Ratio.

Nabil et al. [57] explore the performance of SB-SPS in C-V2X Mode 4, a communication standard designed for V2V communications without relying on infrastructure support. The authors developed an open-source C-V2X simulator, implemented

within the NS-3, and used it to evaluate how different SB-SPS parameters-such as the resource pool configuration, RRI, and the probability of resource reselection-affect the system's performance. Their results indicate that the RRI significantly impacts the Packet Delivery Ratio (PDR). Specifically, increasing the interval from 100 ms to 1000 ms enhanced PDR from 25-55% to 75-90%, although this improvement came at the cost of data rate and increased packet delay. Their findings also reveal that the probability of resource reselection had little effect on performance in dense traffic scenarios, suggesting that the current SB-SPS configuration might not fully exploit the available resources in high-density environments.

Molina-Masegosa, Gozalvez, and Sepulcre [58] analyze several configurable parameters, including the probability P of maintaining previously selected resources, the Sensing Window size, the RSRP threshold, and the vehicle transmit power. Their results show that increasing the probability P, which controls how long vehicles retain selected resources, improves the PDR when the channel load is moderate. However, in highly congested environments, increasing P can lead to higher interference and reduced performance. The study also identifies the critical role of the Sensing Window and RSRP threshold in managing packet collisions and interference in dense vehicular networks.

Bartoletti et al. [59] focus on the mismatch that can occur between the packet generation frequency in the upper layers and the resource allocation periodicity in the physical layer. They explore how the misalignment between these two intervals, particularly the generation interval of CAMs and the SB-SPS allocation period, can affect packet reception and system performance. Their simulations demonstrate that system performance improves when the smallest allocation interval is used and resources are maintained even when no packets are ready for transmission.

Abanto-Leon et al. [60] explore the challenges of resource allocation , focusing on the issue of subchannel selection in Mode 4. The paper introduces an enhancement to the Mode 4 subchannel selection process by applying an Exponentially-Weighted Moving Average to the received power measurements. In the traditional method, power averaging is done linearly over a monitoring window, which gives equal weight to all received power samples. In contrast, the proposed method gives higher priority to the most recent measurements, ensuring that vehicles select subchannels based on the latest interference conditions.

Zhao et al. [61] introduce a cluster-based resource selection scheme for NR-V2X communication, specifically targeting the challenges in LTE-V2X Mode 4. The proposed scheme mitigates the collisions by dividing available resources into different resource sets and assigning these sets to clusters of vehicles. The cluster head, acting as a coordinator, selects the appropriate resource set based on energy measurements and interference levels. The cluster-based approach involves dividing resources into orthogonal sets to reduce interference between clusters. Each vehicle determines whether it will act as a cluster head or a member based on sensing. The cluster head is responsible for selecting and reserving resources within its cluster, thus improving resource management and reducing the probability of resource collision. The improvement is especially noticeable in highway scenarios, where the proposed cluster-based approach provides up to 10% better performance in Packet Reception

Ratio (PRR) compared to Mode 4.

Molina-Masegosa, Sepulcre, and Gozalvez [62] introduce a novel geo-based scheduling scheme that enables vehicles to select radio resources based on their location and the order of neighboring vehicles on the road. By using geographical information from CAMs, the scheme allows vehicles to implicitly coordinate their resource selection, even with those vehicles outside the sensing range. This reduces packet collisions and improves the overall performance of Mode 4. Through both analytical and simulation-based evaluations, the proposed geo-based scheduling scheme demonstrates reduction in packet collisions and an increase in performance compared to the SB-SPS scheme.

2.4.2 Research problem related works

Bazzi et al. address the issue of Wireless Blind Spots (WBSs) in Mode 4 in [63]. In C-V2X, vehicles use the SB-SPS method to allocate resources for periodic safety messages. However, imperfect sensing caused by hidden vehicles and half-duplex devices can lead to the selection of conflicting resources, resulting in packet loss and creating WBSs. These WBSs pose a threat to road safety, as vehicles might lose track of each other's position, especially when overtaking or changing lanes. The study analytically characterizes the probability of WBSs occurrences and proposes an enhancement to the legacy autonomous mode to reduce the duration of these blind spots. The proposed solution involves introducing a parameter that limits the number of consecutive periods in which a vehicle can maintain the same resource allocation. This ensures that even if a resource selection error occurs, the duration of its impact is minimized. The proposed protocol achieves a better trade-off between the reliability of beacon messages and the reduction of WBSs.

Jeon and Kim propose an enhancement to the standard SB-SPS algorithm for Mode 4, focusing on reducing packet collisions, improving reliability, and meeting the stringent low-latency and high-reliability requirements in [64]. They introduce a reservation mechanism that explicitly announces the resource to be used for the next transmission, which significantly reduces the uncertainty inherent in the original SB-SPS approach. In the standard SB-SPS, vehicles select resources randomly and without sharing their decisions with other vehicles, which can result in collisions. The new reservation mechanism allows each vehicle to announce its resource selection ahead of time, thus informing neighboring vehicles of its planned transmission. This proactive announcement helps to avoid conflicts, particularly in dense or fringe communication areas, and leads to better resource utilization. The proposed scheme, referred to as Gapped Reservation-Augmented SPS (GRaSPS), shows a significant improvement over traditional SB-SPS in terms of PRR and packet inter-reception time (PIR). The simulation results reveal that the GRaSPS approach outperforms SB-SPS in high-congestion scenarios, particularly at the edge of the communication range where signal power becomes less reliable.

He et al. address the issue of packet collisions in Mode 4, particularly for aperiodic traffic, by proposing a Short Term Sensing-Based Resource Selection (STS-RS)

scheme in [65]. The proposed STS-RS scheme aims to reduce resource contention and packet collisions by introducing a short sensing duration at the beginning of the resource unit. This allows vehicles to make more informed decisions about which resources to select based on sensing results. The STS-RS scheme operates autonomously, without requiring additional signaling overhead or assistance from other radio technologies, making it particularly suitable for distributed C-V2X communications.

Sabeeh et al. [66] investigate decentralized resource selection, particularly focusing on the improvement of packet collision ratios and PRR in Mode 4. This mode relies on autonomous resource selection, where vehicles independently select their radio resources for broadcasting messages. The paper highlights the limitations of the existing SB-SPS, such as its vulnerability to packet collisions, hidden terminal effects, and half-duplex errors. To address these issues, the authors propose an algorithm called Estimation and Reservation Resource Allocation (ERRA), which enhances the performance of resource selection in Mode 4. ERRA improves the efficiency of resource utilization by allowing vehicles to estimate which resource will be free in the next Selection Window. This is achieved by maintaining a list of resource locations associated with a random counter value. The algorithm improves the packet collision ratio and enhances the packet reception ratio, particularly in scenarios with high vehicle density.

Sabeeh and Wesolowski analyze the limitations of the SB-SPS algorithm, which suffers from packet collisions, especially in scenarios with high vehicle density and varying vehicle mobility in [67]. This issue arises when multiple vehicles in the same broadcast range select the same resource allocation. To address these challenges, the authors propose an enhanced version of the Extended-Estimation and Reservation Resource Allocation (E-ERRA) algorithm. E-ERRA builds upon the original ERRA algorithm by incorporating the tracking of pre-reserved resources, allowing vehicles to handle the issue of resource loss when a vehicle exits or enters the awareness range. By maintaining a list of alternative resources, E-ERRA ensures that packet collisions are minimized and resource availability is tracked more reliably. The results show that E-ERRA provides higher reliability and stability in resource allocation, even as the number of vehicles increases.

Yang et al. [68] propose a new scheme called PRESS (Predictive Assessment of Resource Usage) to address the limitations of resource allocation in Mode 4. The PRESS scheme aims to predict future resource usage by leveraging aggregate reselection counter information. By assessing the channel usage for future transmissions, PRESS allows vehicles to select the least used resources with higher accuracy, reducing the likelihood of collisions. The scheme improves the PRR and enhances resource utilization by more accurately assessing the channel state. This is particularly useful for environments with high vehicle density, where accurate resource usage predictions are crucial for minimizing collisions. Through system-level simulations in various scenarios, including crossroad and highway conditions, PRESS demonstrated significant performance improvements over legacy schemes.

One of the studies that contributed significantly to understanding the performance of the C-V2X standard was by Molina et al. [56]. This work outlines the C-V2X

standard's scheduling mechanism, discusses suggested congestion control mechanisms, and provides an initial evaluation using an internal model that was not made publicly available. The use of this unpublished model motivated the development of the OpenCV2X model [69], enabling academics to replicate the work and explore other aspects of the standard. Later, the same research group released an analytical model of the C-V2X Mode 4 standard, which is used to validate the OpenCV2X model [24]. Additionally, they conducted several studies on C-V2X [58, 26], focusing on the parameterization of SB-SPS, such as RSRP thresholds, $probResourceKeep$, and Sensing Window length, offering valuable guidance on their configuration. These studies also compared the performance of SB-SPS with random scheduling mechanisms and examined the impact of retransmissions.

The most comprehensive study investigating the performance of C-V2X Mode 4 was conducted by Bazzi et al. [25]. This study provides an in-depth analysis of the primary SB-SPS parameters across both the MAC and PHY layers, similar to the work by Molina et al. However, Bazzi et al.'s work is more extensive, examining several realistic environments and offering a comparison with ITS-G5 and random scheduling. The key outcome of this study is the proposed configuration for SB-SPS parameters, including RC, $probResourceKeep$, RSRP thresholds, Sensing Window length, and the proportion of resources reported to the MAC layer after RSSI filtering. It also presents an analysis of the trade-offs involved in selecting various configurations.

As NR-V2X is a relatively recent standard, fewer studies have examined its performance. However, a key study providing a comprehensive analysis of the Release 16 standard can be found in [70]. This study offers an in-depth examination of the changes introduced in NR-V2X Mode 2 from C-V2X Mode 4, with a particular focus on scheduling. It also covers other aspects such as the introduction of unicast and groupcast communications and the modifications made to the SCI format to support these communication paradigms.

Another valuable study by Bazzi et al. [71] provides a thorough literature review in the area, highlighting the various modeling environments, the main challenges faced by C-V2X and NR-V2X, congestion control issues, and the handling of aperiodic traffic patterns. It also discusses some of the key changes made from C-V2X to NR-V2X.

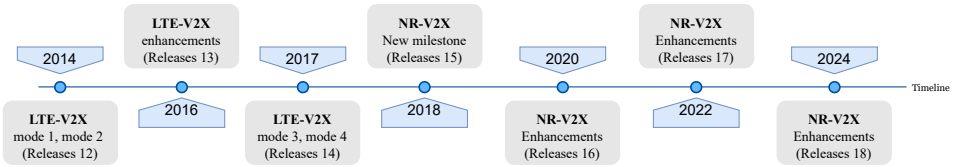


Figure 2.9: Timeline of cellular evolution and how V2X evolves accordingly

2.4.3 C-V2X communications for Platooning

Boubakri & Mettali Gammar [72] provide a survey on intra-platoon communication in autonomous vehicles. They explore the challenges of maintaining stable platoon communication, including latency, safety, and efficiency. The authors discuss various communication technologies, including V2V, V2I, and C-V2X, and propose future research directions to enhance platoon stability and inter-vehicle communication.

Chai et al. [73] focus on platoon partition, power control, and resource allocation for ultra-reliable V2X networks. They propose a two-stage wireless resource allocation algorithm that incorporates power control and spectrum matching. The platoon is divided into multicast clusters, and the system dynamically adjusts resource allocation to improve spectrum efficiency and ensure reliable intra-platoon communication.

Annu and Rajalakshmi [74] propose a novel distance-based queue model for C-V2X platooning communication, focusing on resource allocation optimization during the transition from 5G to 6G. The study integrates spatial considerations into the SB-SPS algorithm, revealing how the distance between vehicles and queue dynamics affect resource allocation efficiency in platooning scenarios.

Chelli et al. [75] explore the performance of NR-V2X for CACC platooning, analyzing the stability of platoons during communication failures and acceleration perturbations. Their findings highlight that the NR-V2X DS scheduling scheme enables shorter inter-vehicle gaps without compromising platoon stability, demonstrating a trade-off between road capacity and energy consumption.

Giambene et al. [76] analyze V2V communications in platooning applications using the LTE-V2X standard as defined in 3rd Generation Partnership Project (3GPP) Release 14. The paper provides an analytical model for CAMs delivery, accounting for V2V link outage probability and the message loss probability due to retransmissions, considering shadowing correlation effects. The model is validated using simulations, offering insights into how to improve platoon communication efficiency in LTE-V2X networks. The paper highlights the use of CACC for maintaining small inter-vehicle gaps in platoons. The platoon leader sends CAMs messages to all platoon members, with a second round of communications from platoon members to update the vehicles behind them. The main challenge addressed is the need for reliable CAMs delivery, especially for the last vehicle in the platoon, where interference from Cellular User Equipment (CUEs) within the same cell can degrade V2V communication.

Anis Boubakri and Sonia Mettali Gammar [77] develop a neuro-fuzzy system that integrates environmental data shared via V2X to optimize the speed and direction of vehicles within a platoon. This approach aims to enhance both energy efficiency and temporal coordination between vehicles. The study demonstrates that their method outperforms traditional techniques, offering potential advancements in platooning technology. This research aligns with ongoing efforts to improve autonomous vehicle coordination and safety through advanced communication and control systems.

Kim et al. [78] explore the challenges of groupcast communication in vehicle pla-

tooning within a V2V network, focusing on the efficiency of retransmissions and resource allocation for communication in platoons. The groupcast mechanism in V2X networks is used to distribute messages from a Platoon Leader to multiple Platoon Members.

Gu et al. [79] address the challenges of providing high reliability and low latency communication in vehicle platooning using the C-V2X network. The authors focus on improving the existing SB-SPS protocol by proposing a Coordinating Resource Reservation (CRR) protocol to reduce collisions and delays caused by conflicting resource reservations.

Segata et al. [80] critically assess the resource allocation schemes in C-V2X networks, particularly focusing on their impact on platooning applications. The study investigates the performance of OoC modes, specifically LTE-V2X Mode 4 and 5G-V2X Mode 2, for vehicle platooning. The authors argue that while C-V2X promises to enhance communication for cooperative driving, certain flaws in its resource allocation mechanisms, especially in OoC scenarios, could hinder its effectiveness in platooning applications. The study focuses on SB-SPS, the scheduling protocol, which is subject to issues like resource reservation conflicts and access collisions. These challenges can lead to long delays and packet loss, which are particularly problematic for platooning applications that require low-latency communication and high reliability. The authors critique the SB-SPS mechanism, noting that it can lead to burst errors and inter-reception delays, which are detrimental to the performance of cooperative driving systems. In addition, the paper identifies burst errors as a significant issue in C-V2X OoC mode, where errors are not uniformly distributed and can cause long sequences of lost packets, which severely affect the stability and safety of vehicle platooning systems. The authors emphasize that the lack of feedback in broadcast communications leads to this issue, as vehicles are unable to adjust their resource allocation in response to the transmission quality.

Wang et al. [81] propose an enhanced resource selection scheme for Mode 4 to support reliable intra-platoon message delivery in Connected and Automated Vehicle (CAV) platoons. The paper introduces several mechanisms to address the challenges of merging collisions and hidden-node collisions. The proposed mechanism divides the frequency-time resources into two sets for vehicles moving in opposite directions, which helps avoid merging collisions between vehicles traveling towards each other on the same road.

Villamil et al. [82] propose a methodology to optimize the packet transmission rates for platooning systems under random access C-V2X communication schemes. The study builds on the Communications-Control Co-Design (CoCoCo) framework, which aims to optimize the usage of communication resources without compromising the quality of control in platooning applications. The key challenge addressed is how to ensure string stability, which prevents oscillations from being amplified through the platoon as vehicles follow each other. The paper proposes a method to calculate the optimal packet transmission rates using Markov Jump Linear Systems. The goal is to determine the Maximum Allowable Packet Loss Probability that ensures the stability of platoon while using the least amount of communication resources. In addition, the work compares random access transmission schemes with

periodic transmission schemes. The results indicate that random access, while providing more flexibility, can lead to packet losses that are detrimental to the string stability if not properly managed. The proposed approach calculates the minimum transmission rate needed for string stability and ensures that the communication system uses resources efficiently.

Wu et al. [83] propose a vehicle selection method for Federated Edge Learning (FEEL) systems in Mode 4 to enhance the accuracy of the model and ensure cache queue stability at the Road Side Units (RSUs). This work aims to optimize the process of selecting which vehicles should upload data for training, based on various system status parameters. The method balances between selecting enough vehicles for accurate model training and ensuring the stability of the RSU's cache queue to avoid overflow.

Mandal et al. [84] investigate the performance of different radio resource allocation algorithms to enhance communication in vehicle platoons supported by 5G enhanced Vehicle-to-Everything (eV2X) communication. The paper compares three platoon Information Flow Topologies (IFTs) - Car-to-Server, Multi-Hop, and One-Hop - for 5G eV2X platooning communications. In each of these topologies, the resource allocation algorithm plays a crucial role in maintaining the stringent latency and reliability requirements. The authors analyze three resource allocation strategies: Maximum Carrier to Interference Ratio (MaxC/I), Proportional Fair, and Deficit Round Robin. The paper also highlights the challenges of maintaining low-latency and high-reliability communication as the platoon size increases. It shows that longer platoons lead to higher communication delays, and as the number of platoons increases in the network, the system faces additional complexities in managing communication effectively.

Wang et al. [85] address the challenge of ensuring reliable intra-platoon communication in CAV platoons, which is critical for safety-related message delivery. Intra-platoon communication faces two main types of packet collisions: hidden-node collisions and merging collisions. To mitigate these issues, the paper proposes a Packet Collision Avoidance Resource Selection (PCA-RS) scheme, enhancing the existing SB-SPS scheme. The PCA-RS scheme introduces three key mechanisms to improve reliability:

- **Resource Partition Mechanism:** Divides frequency-time resources in a Selection Window into two sets for vehicles moving in opposite directions, preventing merging collisions between platoon and non-platoon vehicles.
- **Intra-Platoon Cooperative Mechanism:** Allows the platoon leader to obtain resource occupation information from the last platoon member to avoid hidden-node collisions within the platoon.
- **Merging Collision Detection Mechanism:** Enables non-platoon vehicles to detect the frequency-time resources they occupy after changing lanes, thus preventing merging collisions with other non-platoon vehicles

Shao et al. [86] propose a novel approach to resource allocation in C-V2X platooning systems, called Semantic-Aware Multi-Agent Reinforcement Learning (SAMRAMARL).

The framework integrates semantic communication with multi-agent reinforcement learning (MARL) to optimize resource allocation in dynamic platooning environments. The key idea is to prioritize the transmission of critical semantic information based on the context and importance of the data being shared, improving the overall communication efficiency and quality of experience. The SAMRAMARL approach introduces a distributed learning model where each vehicle (acting as an agent) makes autonomous decisions based on local observations, allowing efficient communication without centralized control. This method optimizes multiple aspects of resource allocation, such as channel assignment, power allocation, and the semantic symbol length, which directly impacts communication efficiency within platoons. By leveraging semantic information, the system focuses on transmitting only meaningful data, thus reducing unnecessary data transfer and improving decision-making in real-time platoon operations.

Sanada et al. [87] analyze the delay distribution of multi-hop relay transmission in a platoon with C-V2X sidelink communication. They show that the single-hop transmission delay in SB-SPS follows a uniform distribution, determined by a single parameter in the scheduling. For multi-hop relay transmissions, the delay distribution is obtained by convoluting the delays of each vehicle in the platoon. Their analytical results are validated through simulations.

Similar approaches are discussed in [87, 88, 89, 90, 91, 92, 93, 94, 95].

2.4.4 Simulation approaches and platforms

There are various approaches to investigating communication technologies, ranging from empirical testing in real-world environments to lower-layer signal modeling and analytical modeling. A common method is simulation, where a technology is implemented to replicate its real-world performance. Simulations can focus on different levels, from full-stack, whole-network simulations that provide an overview of system performance, to link-level simulations that concentrate on PHY and MAC layer performance. These approaches differ in their requirements and focus areas, depending on the objectives of the simulation. Several platforms are available for these simulations, each specialized in specific types, such as full-stack network simulation.

Discrete event simulation models the state of a system on an event-to-event basis, focusing on how the system changes or reacts to these events. In networking, these events typically represent packet generation and may simulate various communication phenomena, such as signal degradation due to path loss or interference. Discrete event simulation is particularly valuable for studying vehicular communication systems, as it allows for the creation of realistic models that capture the complex interactions between vehicles, infrastructure, and communication protocols. By modeling discrete events, interactions, and behaviors, it provides an accurate representation of vehicular communication systems, which is crucial to assessing their performance in dynamic environments. Vehicular communication systems are inherently event-driven, meaning that actions and events occur at specific points in

time rather than continuously. Discrete event simulation is ideal for modeling such systems, as it effectively captures the sporadic nature of vehicular communications and their time-dependent interactions.

The following diagram in Fig. 2.10 provides an overview of the key dates and features available in each model.

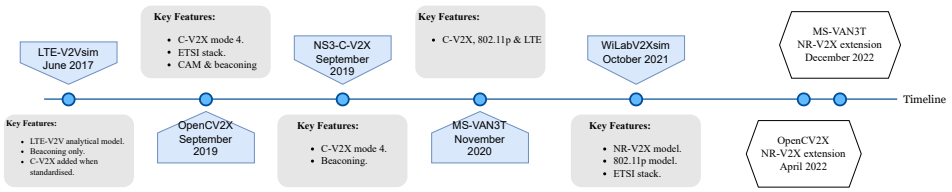


Figure 2.10: Timeline for each model

In the context of C-V2X, two models are available, one serving as a benchmark for validation in this thesis. The first model, LTE-V2V-Sim [96], was based on the C-V2X standard and was later extended and renamed WiLabV2Xsim [29] to incorporate NR-V2X features from Release 16. This model has been widely adopted and offers valuable features, such as enabling simultaneous ITS-G5 and C-V2X/NR-V2X simulation within the same environment. Several of the academic studies mentioned earlier have used LTE-V2V-Sim or WiLabV2Xsim [97, 59].

2.5 Conclusion: C-V2X

To conclude this section of the chapter, it is important to briefly summarize the key areas discussed:

- An introduction to vehicular networking, focusing on regulatory spectrum allocations and ”Day 1” services.
- A summary of the key challenges and advancements in C-V2X communication, particularly in the context of vehicular platooning and the need for efficient resource allocation.
- An overview of the evolution of C-V2X, from its early LTE-V2X standards to the latest NR-V2X advancements, highlighting their impact on communication reliability and latency for vehicular safety and traffic management.
- A discussion of resource allocation mechanisms in C-V2X, including the role of SB-SPS for scheduling and its effectiveness in minimizing packet collisions and ensuring high reliability in high-density traffic scenarios.
- An assessment of the trade-offs in C-V2X’s scheduling configurations, focusing on how parameters such as RRI, Sensing Window, and resource reservation impact performance under varying vehicular network conditions.

- The conclusion on the future of C-V2X, considering ongoing research and improvements, such as enhanced resource selection, the integration of NR-V2X for ultra-reliable low-latency communications, and the potential for widespread deployment in vehicle platooning and autonomous driving.

This background provides a foundation for understanding the subsequent chapters, which will focus on the scheduling mechanism in C-V2X. The first contribution of this work, which will be detailed in chapter 3, involves improving the SB-SPS scheduling mechanism in the platooning environment.

2.6 Background: LoRaWAN

2.6.1 Introduction

This section introduces the foundational concepts and key components of the LoRaWAN technology, which plays a crucial role in enabling long-range, low-power communication for IoT devices. LoRaWAN's architecture, operational principles, and functionalities will be discussed in detail, providing a solid understanding of its role in IoT networks. The subsequent sections will cover the critical aspects of LoRaWAN, including its system architecture, the role of end-nodes and gateways, network management, and the communication protocol stack. Additionally, performance optimization strategies and security mechanisms within LoRaWAN will be explored. The section will conclude with practical guidelines for successfully deploying LoRaWAN-based solutions across diverse IoT applications, addressing challenges such as energy efficiency, scalability, and data integrity. This comprehensive overview will serve as a foundation for understanding how LoRaWAN facilitates efficient and reliable communication in resource-constrained environments.

2.6.2 Description and Definitions of LoRaWAN system

Understanding the architecture of LoRaWAN is of paramount importance for exploring the virtualization of its network components. LoRaWAN stands for the Long-Range Wide-Area Network [98]. It is standardized by the LoRa alliance, which manages the specifications and certification procedures but relies on a proprietary radio technology, named LoRa and patented by Semtech. As suggested by the name, it is a protocol for long-range communications, but operating on low power. Accordingly, LoRaWAN is typically used for IoT-like applications, where the traffic is mainly from the field towards the users (the uplink direction), and devices have several constraints, in particular limited resources in terms of available power. This section delineates the core components of the LoRaWAN architecture: the end node, the gateway, and the backend. As a matter of fact, there is no network layer in the wireless part of a LoRaWAN infrastructure, due to the simple star-of-stars topology (i.e., there is no need for path discovery and routing). Two tiers can be identified:

1. The wireless tier, which includes the end devices, located a single hop away from the gateway (the center of the star);

2. The backend tier, further defined below, which concentrates most of the computational resources and it is often offered as a Platform as a Service (PaaS) in the cloud; it acts as a collector for the data from/to the gateways, thus forming another star consisting of wired (and wireless) links, generally based on the Internet (or Intranet for on-premise solutions).

LoRaWAN Architecture

LoRaWAN operates on a star-of-star topology [99], consisting of three key components: Backend, Gateways (GWs), and End Nodes (ENs). In this architecture, ENs communicate with nearby GWs and each Gateway (GW) is connected to the backend. LoRaWAN networks use an ALOHA based protocol, so ENs don't need to peer with specific GWs. Messages sent from ENs travel through all GWs within range. These messages are received by Network Server (NS) in the backend. If the NS has received multiple copies of the same message, it keeps a single copy of the message and discards others. This is known as message deduplication. Only LoRaWAN GWs have the ability to forward data packets from the ENs to the NS, encapsulating them into UDP/IP packets. A single NS can support multiple Application Server (AS)s which is responsible for securely processing the application data. The general LoRaWAN architecture is illustrated in Fig. 2.11.

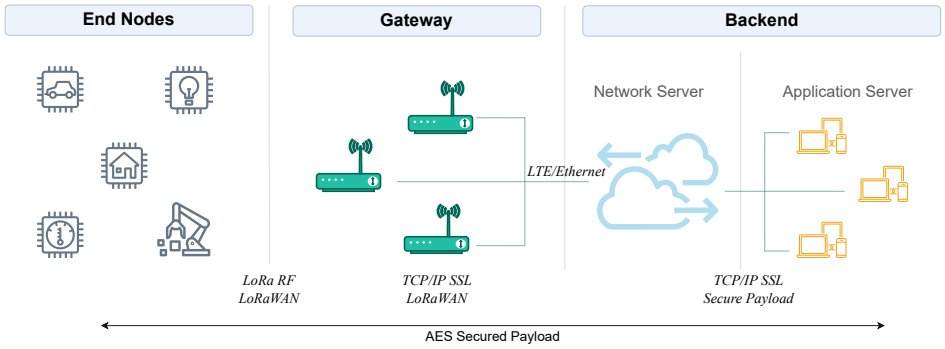


Figure 2.11: LoRaWAN Architecture

End-Node

Within the LoRaWAN architecture, the End Node (EN) operates hosts the sensor or device endpoint that interacts with the physical world and transmits (refined) data to the final sink through one or more gateways. These ENs are typically battery-operated and utilize the LoRa radio frequency modulation technique to transmit small amounts of data over long distances, often in remote or urban environments. Due to their energy-efficient design, LoRaWAN ENs are particularly suited for applications requiring long-range connectivity with minimal power consumption.

In our research, we used the RN2483 to implement and evaluate LoRaWAN-based ENs, investigating their performance (you can see in Fig. 2.12.). The RN2483 module, manufactured by Microchip Technology, is a prominent example of a LoRaWAN-compatible EN frequently employed in IoT applications. It integrates the LoRa radio frequency modulation technology with embedded protocol management capabilities, allowing seamless and efficient communication within LoRaWAN networks.

The RN2483 operates in the sub-GHz frequency band (specifically 868 MHz in Europe), enabling extended communication ranges with minimal power consumption, thus making it highly suitable for battery-operated sensor nodes and IoT deployments in challenging or remote environments.

Figure 2.12: RN2483, LoRaWAN end node

Gateway

Gateways act as the communication bridge between end nodes and the backend server. They receive frames from multiple co-located end nodes and relay this information to the backend. Unlike end nodes, which operate on configurable but fixed channels and SFs, gateways are designed to concurrently receive traffic by leveraging different channels and SFs, thus minimizing the impact of the limited bandwidth and the simple MAC strategy. As regards the connectivity towards the backend, the specifications do not define how it is actually implemented; however, they are typically connected to an IP-based wired or wireless infrastructure. In particular, the GW executes the so-called packet forwarder software, in charge of embedding/extracting LoRaWAN payloads into the payload of the protocol connecting with the backend. Currently, two main packet forwarders exist:

1. The UDP packet forwarder, also referred to as the legacy packet forwarder, is the original implementation, initially proposed by Semtech. It uses UDP as the transport protocol to transfer LoRaWAN packets. This solution is widely adopted in both industry and academia, and most manufacturers preinstall it on their gateways. The protocol is quite simple, facilitating the implementation, but also has some drawbacks. Notably, the link between the gateway and the backend is not encrypted, and the use of the unreliable UDP protocol

can lead to message loss. However, these drawbacks are overcome using VPN tunneling, which ensures both privacy and reliability.

2. The LoRa Basics Station uses the WebSockets technology to communicate, natively supporting data encryption as it moves between the gateway and the network server. A two-step procedure is supported, in which a preliminary discovery procedure occurs before the actual connection with the gateway; additionally, a Configuration and Update Server (CUPS) is defined, which allows for easy management of device update and configuration.

It is also interesting to note that frontend solutions, like the Semtech SX125X devices, tailored for the design of LoRaWAN Gateways, allow us to lower the cost of the hardware for SDR-based LoRaWAN transceivers [100], which is relevant for medium- and large-scale deployments. Moreover, these devices also have limited supply currents, especially in standby and sleep modes, a key characteristic to limit overall power consumption.

Figure 2.13 provides an example of a LoRaWAN gateway, specifically the RG168 Laird Ethernet LoRaWAN Gateway.

Figure 2.13: RG168, Laird Ethernet LoRaWAN Gateway

Backend

The backend of a LoRaWAN network is the core of operations, comprising several key entities that manage the flow of data and ensure secure and efficient network functionality. These entities include the NS, AS, and Join Server (JS):

- *NS*: The NS manages the routing of data across the network, ensuring efficient message delivery by deduplicating messages from multiple gateways and

maintaining data integrity through network session keys. It also handles network responses such as acknowledgments and adaptive data rate commands. The NS is the endpoint of the packet forwarder connection.

- ***AS***: The AS interfaces with user applications, providing secure data delivery by decrypting messages using application-specific keys. It tailors data processing to meet the specific needs of various applications.
- ***JS***: The JS enhances network security by managing the authentication and initial device onboarding processes. It securely handles device root keys and session key generation, ensuring that each device's identity is secure and unique.

Additional details can be found in [101].

In this study, The Things Network (TTN) is utilized as the backend infrastructure for the LoRaWAN network. TTN is an open-source, decentralized LPWAN designed to connect IoT devices through the LoRaWAN protocol [102]. It provides an accessible, cost-effective infrastructure for IoT communications on a global scale, enabling community-driven deployments via locally installed gateways. TTN plays a central role in connecting IoT devices to applications that process and analyze the collected data [103, 104].

TTN's architecture is composed of several key components that collaborate to handle the communication between ENs and AS. These components include the gateways, routers, broker, handler, and AS, as illustrated in Fig. 2.14 of the document. The diagram provides a detailed overview of the architecture and data flow within TTN.

The diagram above shows the main components of the TTN network and how they interact with each other. The TTN architecture is based on several key entities:

- *Packet Forwarder & Gateway Connector:* The packet forwarders receive messages from the EN and relay them to the network. These packet forwarders as explained in sec. 2.6.2 are typically hosted on the gateway hardware. The gateway connectors are responsible for forwarding the messages between the gateways and the network infrastructure.
- *Router (R):* The router is responsible for managing the routing of messages between gateways and the TTN network. It ensures that messages from the devices are passed on to the correct components for further processing. It also manages the communication between the gateways and the TTN backend.
- *Network Server (NS):* The NS handles the de-duplication of messages, the management of network keys, and the adaptation of messages to the correct format. It also performs crucial tasks such as ensuring data integrity and securing communications between the devices and the network infrastructure.
- *Broker (B):* The broker serves as the central point for message forwarding. It handles the reception of messages from the router and forwards them to the appropriate handler. The broker also manages the de-duplication of messages

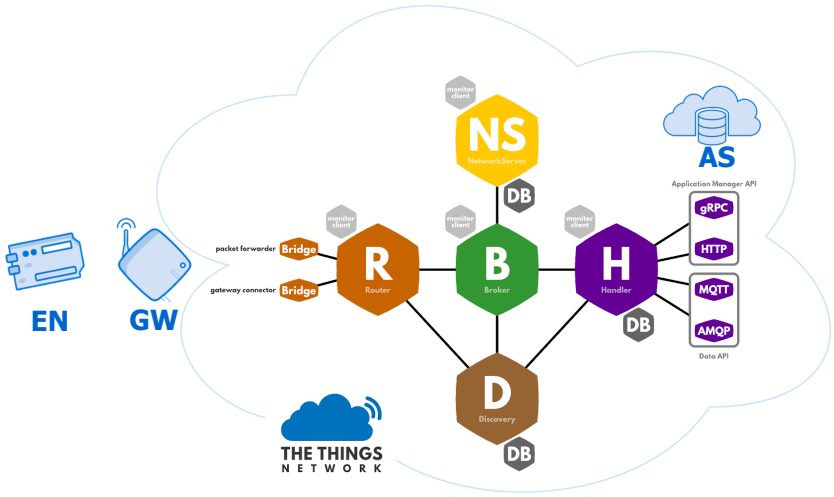


Figure 2.14: Architecture in TTN

and performs necessary cryptographic checks to ensure the authenticity and integrity of the messages.

- *Handler (H):* The handler is responsible for decrypting the data and forwarding it to the relevant AS for further processing. It also handles the encryption of downlink messages and ensures that they are correctly formatted before sending them to the ENs.

- *Discovery (D):* This component is responsible for discovering and managing the configuration of network components, ensuring that new devices or gateways can be registered and managed efficiently.

The architecture also includes communication with external systems via different protocols. The Data API allows applications to interact with the network and devices through various communication protocols, including gRPC, HTTP [1], MQTT, and AMQP [2].

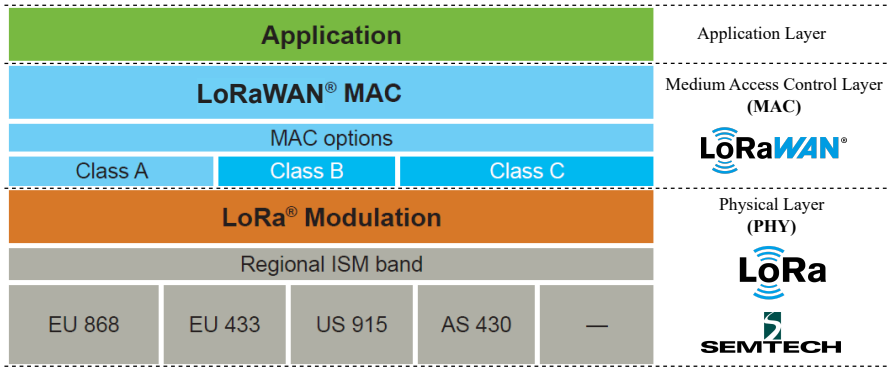


Figure 2.15: Stack structure in LoRaWAN protocol

2.6.3 End Node Services

The functionalities and services provided by the EN protocol stack can be classified into three primary layers: Application, MAC, and PHY, as illustrated in Figure 2.15. Each layer encapsulates data progressively, adding specific control information as the payload traverses downward through the stack. Detailed descriptions of the MAC and PHY layers, due to their significance in this thesis, are provided separately in Sections 2.6.4 and 2.6.5, respectively. Here, we focus exclusively on the Application Layer.

Application Layer

The Application Layer constitutes the intelligence of a LoRaWAN EN, serving as the central processing unit responsible for handling sensor inputs and orchestrating data transmissions. Its primary functions include:

- **Data Acquisition and Processing:** Gathering raw sensor data (e.g., temperature, humidity, motion) and applying pre-defined logic or threshold criteria to generate meaningful information.
- **Payload Formatting:** Preparing sensor data into compact, efficient payloads suitable for transmission. This may involve encoding, compressing, encrypting, or applying additional protocol-specific wrapping.
- **Transmission Control:** Initiating uplink communications by transmitting payloads to the NS and managing responses to downlink commands, such as device configuration updates or operational instructions.

[1] **gRPC** and **HTTP** are commonly used for high-performance, reliable communication between network components and external applications.

[2] **MQTT** and **AMQP** are lightweight messaging protocols commonly used for IoT communication due to their efficiency in terms of bandwidth and power consumption.

It is important to note that the LoRaWAN specifications intentionally leave payload formatting undefined, providing only a generic binary field within the message structure. Consequently, developers must implement custom payload formatters. Typically provided by real-world LoRaWAN backends, these payload formatters are code snippets responsible for interpreting and converting raw binary payloads into meaningful, human-readable data representations and vice versa.

2.6.4 LoRaWAN® MAC

Overview

The LoRaWAN MAC layer defines how ENs and gateways coordinate access to the shared radio channel, handle addressing, security, and manage NS interactions in a star-of-stars topology. ENs communicate uplinks to one or more gateways, which forward packets over IP to a NS. Downlinks from the NS traverse gateways back to ENs, supporting both configurable receive windows and class-dependent scheduling. In LoRaWAN networks, devices randomly access the wireless channel using the pure ALOHA method, as specified by the LoRaWAN standard, and adhere to a strict duty cycle. In cases where the hidden node problem is taken into account and the listen-before-talk procedure is omitted to conserve energy, the use of the ALOHA MAC protocol is appropriate [105].

Device Classes

The LoRaWAN protocol specification [106, 107, 108] outlines three categories of device operation, referred to as Classes A, B, and C (See Fig. 2.16). Every LoRaWAN-compliant device must support Class A, which forms the foundational mode of operation, while Classes B and C are optional enhancements that build on Class A functionality. Each of these device classes supports two-way communication, meaning they can both send and receive (uplink and downlink) messages.

- **Class A** (All devices):
  - Mandatory for all ENs.
  - Uplink transmissions are followed by two short receive windows (RX1, RX2) at pre-defined delays.
  - Offers the lowest downlink latency only immediately after uplink.
- **Class B** (Beacon-synchronized):
  - Builds on Class A by adding scheduled receive windows synchronized via periodic gateway beacons.
  - Devices open extra ”ping-slots” at known times, enabling the network to send downlinks with bounded latency.
- **Class C** (Continuous):

- Devices keep their second receive window (RX2) open continuously (except when transmitting).
- Provides lowest downlink latency at the cost of higher power consumption.

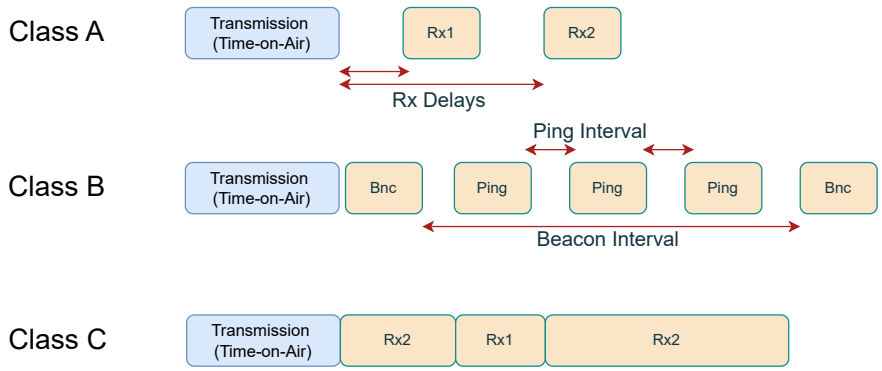


Figure 2.16: Device Classes

Adaptive Data Rate (ADR) Mechanism

Adaptive Data Rate (ADR) is a LoRaWAN feature that dynamically controls each EN's data rate (via Spreading Factor (SF) and bandwidth (BW)) and transmit power to optimize network capacity and minimize airtime and energy consumption [109]. The NS evaluates link quality metrics-primarily the Signal-to-Noise Ratio (SNR) margin, defined as the difference between the measured SNR from the gateway and the minimum required SNR for the current data rate-and uses this margin to decide whether an EN can safely increase its data rate or reduce its transmit power without compromising reliability [110, 111]. When ADR is enabled (the ADR control bit in the FCtrl byte set to 1, as per Section 4.3.1.1 of the LoRaWAN L2 specification [109, 110]), the server sends LinkADRReq MAC commands carrying new Data Rate and Tx-Power parameters. The device applies these settings after returning LinkADRAns acknowledgments, allowing the network to fine-tune each link over successive uplinks. ADR is most effective for stationary or slow-varying links, where channel conditions remain stable. For highly mobile applications-such as asset trackers-the channel can change too quickly for ADR to converge, so ADR is typically disabled in those scenarios to avoid oscillations or suboptimal settings [112, 113].

Joining Procedure

LoRaWAN supports two activation methods-Over-The-Air-Activation (OTAA) and Activation by Personalization (ABP)-each with distinct message flows, security properties, and configuration steps. OTAA provides dynamic device provisioning and

enhanced security through unique session keys and addresses derived during network join, while ABP simplifies deployment with static keys and addresses preloaded on devices, trading off security for ease of commissioning.

1. Over-The-Air Activation (OTAA)

The OTAA process involves an initial handshake between the device and the network server to securely exchange session keys and device addresses. Fig. 2.17 illustrates the complete OTAA message flow according to LoRaWAN 1.1 specifications.

Message Exchange:

$$\underbrace{\text{JoinReq}}_{\substack{\text{MHDR}=0x00 \\ \text{AppEUI} \,\|\, \text{DevEUI} \,\|\, \text{DevNonce}}} \longrightarrow \underbrace{\text{JoinAccept}}_{\substack{\text{MHDR}=0x20 \\ \text{AppNonce} \,\|\, \text{NetID} \,\|\, \text{DevAddr} \,\|\, \text{DLSettings} \,\|\, \text{RxDelay} \,\|\, \text{CFList?}}}$$

(then derive session keys and reset counters).

- **JoinReq (Up-link):**
  - MAC header (MHDR) (1 byte): 0x00 (Join-Request)
  - `AppEUI` (8 bytes, LSB first): 64-bit Application identifier
  - `DevEUI` (8 bytes, LSB first): 64-bit Device identifier
  - `DevNonce` (2 bytes): Random nonce generated by device; must be unique per device reboot to prevent replay.
  - `MIC` (4 bytes): AES-CMAC over MHDR JoinPayload using AppKey.

- **JoinAccept (Down-link):**
  - `MHDR` (1 byte): 0x20 (Join-Accept)
  - Encrypted Payload (12 to 16 bytes):
    - `AppNonce` (3 bytes): Server-generated nonce
    - `NetID` (3 bytes): Network identifier
    - `DevAddr` (4 bytes): Assigned 32-bit device address
    - `DLSettings` (1 byte):
      - Bits 7-4: RFU (set to 0)
      - Bits 3-0: RX2 Data Rate (DR) and RX1DROffset parameters
    - RxDelay (1 byte): RX1 window delay (in seconds)
    - CFList (optional 16 bytes): Up to five 24-bit channel frequencies for EU868 (absent in US915).
  - MIC (4 bytes): AES-CMAC over MHDR and encrypted payload using AppKey.

  - The entire JoinAccept payload (excluding MHDR) is encrypted with AppKey using AES-128 in ECB mode before MIC is appended.

- **Session-Key Derivation**: Using the secret *AppKey*, device and server derive the two AES-128 session keys:

$$\text{NwkSKey} = \text{AES128}_{AppKey}(0x01 \,\|\, \text{AppNonce} \,\|\, \text{NetID} \,\|\, \text{DevNonce} \,\|\, \underbrace{0\ldots0}_{\text{7 bytes}})$$

$$\text{AppSKey} = \text{AES128}_{AppKey}(0x02 \,\|\, \text{AppNonce} \,\|\, \text{NetID} \,\|\, \text{DevNonce} \,\|\, 0\ldots0)$$

  After this, the device resets its uplink and downlink frame counters to zero.

- **Timing and Retries:**
  - JoinReq is sent at the device's lowest data rate (e.g., SF12/BW125) to maximize reachability.
  - The device waits for a JoinAccept in its two receive windows (RX1, RX2), configured by RxDelay and DLSettings.
  - If no valid JoinAccept is received after a configurable number of retries, the device may back off before retrying.

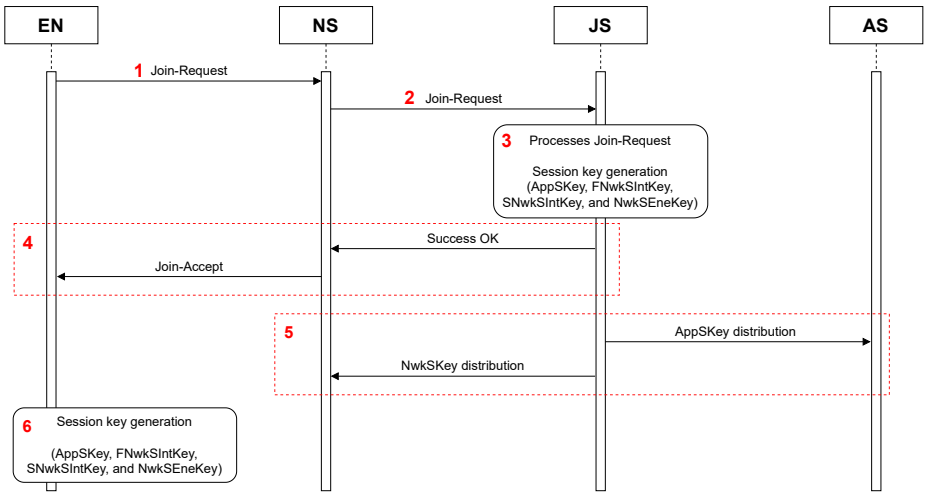


Figure 2.17: OTAA message flow in LoRaWAN 1.1

2. Activation By Personalization (ABP)

ABP bypasses the dynamic join procedure, instead relying on static, pre-shared keys and addresses directly configured into the device. This method eliminates the overhead associated with the join procedure but introduces potential security risks, particularly concerning key reuse and replay vulnerabilities. Figure 2.18 demonstrates how ABP devices operate with pre-shared session parameters.

- Static Configuration:
  - DevAddr, NwkSKey, AppSKey are pre-provisioned in the device (no JoinReq/JoinAccept exchange).
  - Simplifies commissioning but *bypasses* dynamic address assignment and the fresh key-derivation security of OTAA.
- Security Considerations:
  - Since ABP devices reuse fixed keys, they are more vulnerable to replay and brute-force attacks.
  - Frame counters (FCntUp, FCntDown) must be managed carefully: after a power-cycle, resetting counters can expose devices to replay; some implementations store counters in non-volatile memory.

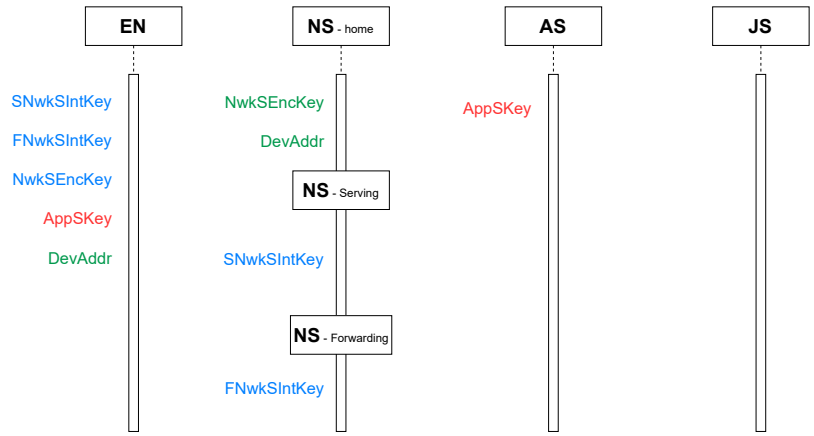


Figure 2.18: Pre-sharing DevAddr and session keys for ABP in LoRaWAN 1.1

MAC Versions

The LoRaWAN® MAC specification has been iteratively refined to add functionality, clarify behavior, and decouple region-specific parameters. Below is an updated, concise summary of the major releases, highlighting key enhancements and pointing to the official Revisions section and tables in the spec: Tab. 2.3 summarizes the major releases, their dates, and key enhancements. For the official change-log and full normative details, refer to Section 21 "Revisions" in the LoRaWAN® Specification v1.0.3 [106] and Section 17 "Revisions" in the LoRaWAN® Specification v1.0.4 [110].

Table 2.3: LoRaWAN® MAC Specification Version History

Version	Release Date	Key Highlights
1.0.0	January 2015	First formal L2 spec: uplink/downlink frame formats, core MAC commands, security (MIC, AES-CMAC), device classes; inline regional parameters.
1.0.1	February 2016	Clarifications and typo fixes: RX window timing, DR2 payload limit (US915), JoinAccept MIC calculation, NbRep→NbTrans rename, coding-rate requirement.
1.0.2	July 2016	Regional parameters extracted to RP002; added `DlChannelRec`; `TxParamSetupReq` MAC commands; fixed ADR back-off logic; batch ADR requests; clarified AppKey usage.
1.1.0	October 2017	Introduced Active Roaming support (HandoverReq/Ans), multicast refinements, Class B beacon scheduling, and enhanced security mechanisms. [108]
1.0.3	July 2018	• Back-ported Class B into 1.0.x • Added DeviceTimeReq/Ans for Class B beacon acquisition • Deprecated BeaconTimingReq/Ans • Corrected GPS epoch references and various typos • Added Over-The-Air Firmware Update (FUOTA) support [106]
1.0.4	October 2020	Terminology update: JoinEUI/JoinNonce replaced AppEUI/AppNonce; additive RX logic for Classes A/B/C; normative clean-up; RP002 cross-reference. [107]
RP002 1.0.x	2016-2021	Standalone Regional Parameters spec: global ISM-band plans, duty-cycle limits, added new regions (e.g., AS923-4, IL915); updated independently.

Frame Format

LoRaWAN MAC messages are carried within the radio PHY payload (PHYPayload), which comprises three fields (Tab.2.4):

$$\text{PHYPayload} = \underbrace{\text{MHDR}}_{\text{1 byte}} \parallel \underbrace{\text{MACPayload}}_{\text{1…M bytes}} \parallel \underbrace{\text{MIC}}_{\text{4 bytes}}$$

Table 2.4: LoRaWAN® MAC Layer Message Format

Field	Size (bytes)	Description
MHDR	1	MAC header: message type and protocol version
MACPayload	1.M	Frame header + optional port + payload (application data or MAC commands)
MIC	4	Message Integrity Code (AES-CMAC over MHDR‖FHDR‖FPort‖FRMPayload)

Note: M is the maximum MACPayload length, equal to the maximum PHY payload (255 bytes) minus MHDR (1 byte) and MIC (4 bytes), so $M_{\max} = 250$ bytes.

MAC Header (MHDR)

The 1-byte MHDR defines how the remainder of the message is interpreted (Fig.2.19).

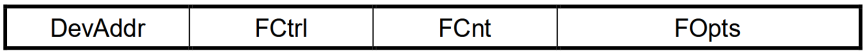


Figure 2.19: Frame header structure

Table 2.5: Frame Header (MHDR) Bit Fields

Bits	Field	Description
7…5	MType	Message type (e.g., Join-Req, Join-Accept, Unconfirmed Data Up/Down, Confirmed Data Up/Down)
4…2	RFU	Reserved (set to 0)
1…0	Major	LoRaWAN major version (0x0 = v1.0)

MACPayload Structure

Fig.2.20 illustrates the MACPayload, which for data messages breaks down as:

Figure 2.20: MAC payload structure

$$\underbrace{\text{FHDR}}_{7\ldots22} \,\|\, \underbrace{\text{FPort}}_{0\ldots1} \,\|\, \underbrace{\text{FRMPayload}}_{0\ldots N}$$

- `FHDR` (Frame Header, 7…22 bytes)
  - `DevAddr` (4 bytes): 32-bit device address
  - `FCtrl` (1 byte): frame-control flags
    - Bit 7: `ADR` (Adaptive Data Rate)
    - Bit 6: `ADRACKReq` (ADR acknowledgment request)
    - Bit 5: `ACK` (frame acknowledgment)
    - Bit 4: `FPending` (downlink pending) or Class B flag in uplink
    - Bits 3…0: `FOptsLen` (length of optional `FOpts` field)
  - `FCnt` (2 bytes): frame counter (uplink or downlink)
  - `FOpts` (`0...15` bytes): optional MAC commands (up to 15 bytes)
- `FPort` (`0...1` byte)
  - `If present and = 0`: FRMPayload contains MAC commands (encrypted)
  - `If > 0`: FRMPayload carries application payload (encrypted)
- FRMPayload (`0...N bytes`)
  - Encrypted using AppSKey (application data) or contains plaintext MAC commands if `FPort=0`

Message Integrity Code (MIC)

The 4-byte MIC is computed over MHDR || FHDR || FPort || FRMPayload using AES-CMAC with the network session key (NwkSKey), as specified in RFC 4493 .

2.6.5 LoRa® PHY

Overview

LoRa (Long Range) is a proprietary physical-layer modulation scheme developed by Semtech that implements CSS to achieve long-range, low-power wireless communication. By encoding symbols as linear up-chirps and down-chirps across a wide bandwidth, LoRa offers high resilience to interference, Doppler shifts, and multipath fading, making it well-suited for IoT deployments in unlicensed ISM bands (e.g., 868 MHz in Europe, 915 MHz in North America).

Modulation

LoRa PHY employs Frequency Shift - Chirp Spread Spectrum (FS-CSS) modulation, where each symbol is represented by a chirp whose starting frequency is offset according to the symbol value. The instantaneous frequency increases (up-chirp) or decreases (down-chirp) linearly at a rate

$$\mu = \frac{B}{T_M},$$

where B is the bandwidth and T_M is the symbol duration. Sampling at $T_s = 1/B$ yields $M = B\,T_M$ samples per symbol, allowing efficient de-chirping and symbol detection via FFT: mixing the received signal with a reference down-chirp produces a near-constant tone whose FFT bin index directly maps to the transmitted symbol.

The time-domain waveform in Fig. 2.21 plots several consecutive LoRa symbols at baseband. Each segment is a constant-amplitude sinusoid whose instantaneous frequency increases linearly over one symbol period T_M (an up-chirp).

In Fig.2.22, the horizontal axis spans one symbol period T_M, while the vertical axis shows instantaneous frequency normalized about the carrier (from $f_{\min} = f_c - \frac{B}{2}$ up to $f_{\max} = f_c + \frac{B}{2}$) .

- The up-chirp trace (solid line) starts at $f_{\min} + f_{offset}$ and increases linearly at rate $\mu = B/T_M$ until it reaches $f_{\max}$.
- The down-chirp (dashed line) sweeps in the opposite direction, providing the reference for de-chirping.
- Different symbol values S simply shift the start of the sweep by $f_{offset} = \frac{B}{2^{SF}}\,S$, which is why you see multiple parallel lines in the figure.

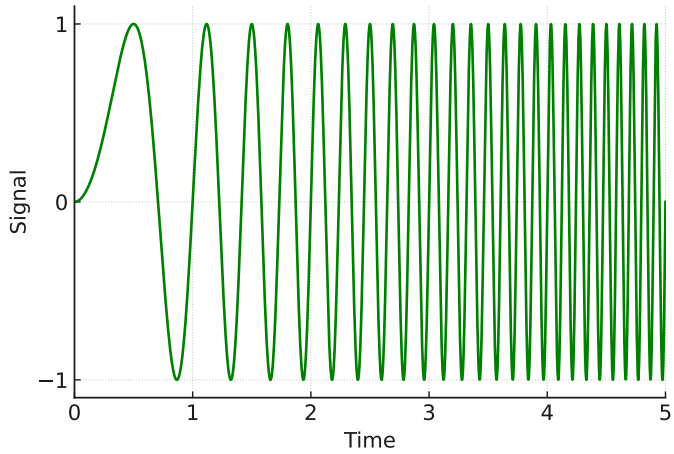


Figure 2.21: LoRa chirp signal

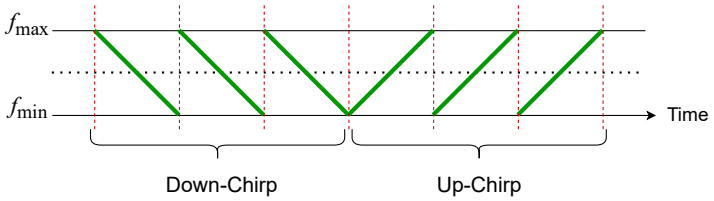


Figure 2.22: LoRa chirp signal

- When the instantaneous frequency would exceed $f_{\max}$, it "folds" back to $f_{\min}$ (due to sampling at $f_s = B$), creating the sawtooth-like waveform. This folding is essential: it guarantees that every symbol chirp has exactly 2^{SF} distinct phase trajectories, all with a constant envelope, which in turn maximizes power-amplifier efficiency and yields the high processing-gain that lets LoRa operate below the noise floor.
- By multiplying the received up-chirp by the conjugate of the reference down-chirp (an operation illustrated by overlaying the two sweeps in the figure), the quadratic phase term cancels, leaving a pure tone whose frequency offset directly equals f_{offset}. An 2^{SF}-point FFT then localizes that tone into one bin at index S, recovering the transmitted symbol with very low complexity.

LoRa Signal

The physical characteristics of a LoRa transmission are defined by a set of key parameters, each of which can be tuned to trade off range, data rate, and robustness. Tab.2.6 below summarizes these parameters and their typical settings. The paragraphs that

follow explain each specification in detail.

Table 2.6: Overview of LoRa transmission parameters

Parameter	Description	Typical Values / Range
CF	Center frequency of each channel	EU868: 868.1, 868.3, 868.5 MHz; US915: 902.3-914.9 MHz in 200 kHz steps
BW	Chirp frequency span	125 kHz, 250 kHz, 500 kHz
SF	Bits encoded per symbol	7, 8, 9, 10, 11, 12
SR	Symbols/s; $R_S = \frac{B}{2^{\mathrm{SF}}}$	SF7 @ 125 kHz: 976.56 sps; SF12 @ 125 kHz: 30.52 sps
T_M	Time per symbol; $T_M = \frac{2^{\mathrm{SF}}}{B}$	SF7 @ 125 kHz: 1.024 ms; SF12 @ 125 kHz: 32.768 ms
CR	FEC ratio, $\frac{4}{4+n}$, $n \in \{1, \ldots, 4\}$	4/5, 4/6, 4/7, 4/8

- *Carrier Frequency (CF)*: The Carrier Frequency (CF) defines the channel's midpoint and is set according to regional ISM regulations. For example, in the EU868 band CF often takes values such as 868.1, 868.3, and 868.5 MHz, while in the US915 band there are 64 uplink channels from 902.3 MHz incremented by 200 kHz up to 914.9 MHz

- *Bandwidth (BW)*: BW determines the frequency span of each chirp. Standard LoRa settings are 125 kHz, 250 kHz, and 500 kHz. A wider BW doubles the bit rate for a fixed SF and CR but reduces processing gain and sensitivity, making BW selection a key trade-off between throughput and range.

- *Spreading Factor (SF)*: The SF is the number of bits encoded per symbol, taking integer values from 7 to 12. Higher SF increases the processing gain-improving receiver sensitivity (e.g., from approximately -123 dBm at SF7 to -137 dBm at SF12)-at the cost of lower data rate and longer time-on-air. Orthogonality between different SFs enables concurrent transmissions on the same frequency.

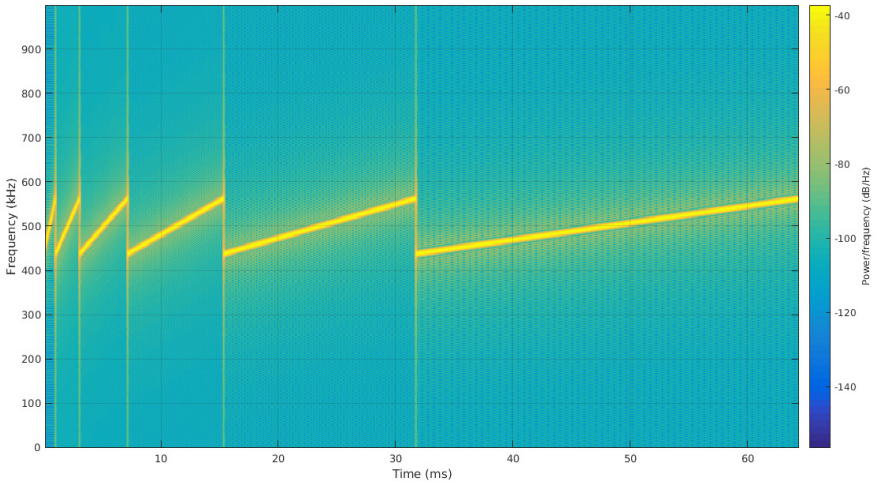


Figure 2.23: Comparasion of LoRa SFs: SF7 to SF12

- *Symbol Rate (SR)*: Symbol rate (R_S) is the rate at which LoRa symbols are transmitted, given by

$$R_S = \frac{B}{2^{\text{SF}}},$$

while the symbol duration is

$$T_M = \frac{2^{\text{SF}}}{B}.$$

Thus, reducing SF or increasing BW raises the symbol rate-and hence throughput-while decreasing processing gain and link budget.

- *Coding Rate (CR)*: Coding Rate (CR) specifies the ratio of payload bits to total bits (payload + FEC) and is defined as

$$\text{CR} = \frac{4}{4+n}, \quad n \in \{1, 2, 3, 4\},$$

yielding CR values of $4/5$, $4/6$, $4/7$, and $4/8$. Higher coding rates add more redundancy to improve robustness against errors and burst interference, but they also increase time-on-air and energy consumption

Frame Format

LoRa PHY supports two header modes-explicit and implicit-which determine whether per-packet header fields are included. The explicit header mode is most common for data transmissions: it embeds length, coding, and integrity-check information directly in the radio packet. Fig.2.24 (Downlink PHY structure) and Fig.2.25 (Radio PHY structure) from the LoRaWAN 1.0.3 spec illustrate this format:

Figure 2.24: Downlink PHY structure

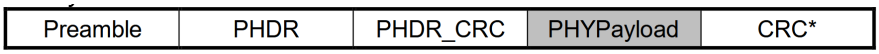


Figure 2.25: Radio PHY structure (CRC* is only available on uplink messages)

- Preamble: A sequence of unmodulated up-chirps used for coarse time and frequency synchronization. LoRaWAN mandates 8 symbols of preamble, but the radio transceiver adds 4.25 start-frame symbols, yielding a default of 12.25 symbols in total.
- Physical Header (PHDR): A one-byte field that encodes:
  1. The length of the PHY payload (in bytes),

2. The coding rate used for FEC,
3. A flag indicating whether a payload Cyclic Redundancy Check (CRC) is present.

This header is encoded using the same coding rate as the payload.

- PHDR_CRC: A 2-symbol CRC protecting the PHDR itself. This error-detection code ensures correct parsing of the payload length and coding-rate information before decoding the main payload.
- PHYPayload: The actual data bits to be transmitted, after whitening and error-correction coding. In LoRaWAN, this is the MAC frame (MHDR + MACPayload + MIC); in proprietary PHY uses it may be arbitrary user data.
- CRC: A 2-symbol CRC over the PHYPayload, only included on uplink messages for end-to-network integrity (omitted on downlinks to save airtime).

In implicit header mode, both PHDR and PHDR_CRC are omitted and no payload CRC is used; the receiver must be pre-configured with the exact payload length and coding rate, which can slightly reduce airtime at the cost of flexibility.

2.6.6 Description and Definitions of Virtualization Technologies

Overview

Virtualization refers to the process of abstracting physical hardware resources to create multiple isolated virtual environments capable of running applications independently. Lightweight virtualization, specifically container-based virtualization, stands out as a highly suitable technology to empower edge computing within IoT environments. Containers encapsulate applications along with their dependencies into portable units, providing consistent execution across diverse hardware and software environments without the substantial overhead typically associated with traditional virtual machines. This technology enables rapid application deployment, scaling, and efficient management, facilitating seamless interaction between cloud and edge nodes, thus enhancing the flexibility and responsiveness of IoT systems.

The use of virtualization in IoT scenarios provides several key advantages:

- *Resource Efficiency*: Containers share the host system's kernel, significantly reducing memory and CPU overhead, making them ideal for resource-constrained edge devices.
- *Portability and Flexibility*: Containerized applications are easily portable across various platforms, ensuring consistent operation irrespective of underlying hardware or software differences.
- *Scalability and Rapid Deployment*: Containers allow quick instantiation, enabling scalable and dynamic responses to changing IoT workloads and service demands.

- *Isolation and Security*: Containers provide robust isolation between applications, enhancing the security and reliability of multi-tenant IoT deployments.

Three primary types of lightweight virtualization technologies (Fig. 2.26) include:

1. **Container-Based Virtualization**: Utilizes shared kernel architecture for lightweight and portable application execution.
2. **Unikernels**: Minimalistic, single-address-space machine images constructed by using specialized libraries, enabling extremely efficient operation.
3. **Virtual Machines (VMs)**: Traditional hypervisor-based virtualization creating completely isolated virtual environments with their own kernels.

VMs like KVM [3] and QEMU [4] rely on a hypervisor to host guest operating systems and their applications, each running on isolated kernels. Unikernels leverage a library operating system tailored specifically for single applications, offering highly optimized execution environments. Containers use a container engine running directly on the host operating system to isolate applications along with their dependencies, efficiently sharing a single kernel and significantly reducing resource usage compared to traditional VMs.

Traditional virtualization like typically requires separate kernels and dedicated resources for each virtual instance, resulting in substantial resource consumption and overhead, which is not suitable for resource-constrained IoT devices [114]. In contrast, lightweight virtualization shares the host operating system's kernel among multiple virtual instances, significantly reducing both memory and CPU overhead [115]. Additionally, lightweight virtualization techniques, such as containers and unikernels, offer rapid startup times, often initiating instances almost instantly, whereas traditional virtual machines have notably longer boot times [116].

Container-Based Virtualization

Containerization encapsulates applications along with their dependencies into self-contained units, providing consistent execution environments that share the host OS kernel, greatly reducing resource requirements.

- **LXC**:

  LXC represent a core containerization technology integrated into Linux, enabling the creation of isolated user-space environments through kernel namespaces and control groups (cgroups). Kernel namespaces offer isolation by providing processes with their own view of system resources, such as filesystems,

[3] Kernel Virtual Machine. Project site: `https://www.linux-kvm.org/page/Main_Page` (accessed April 28, 2025).

[4] A generic and open source machine emulator and virtualizer. Project site: `https://www.qemu.org/` (accessed April 28, 2025).

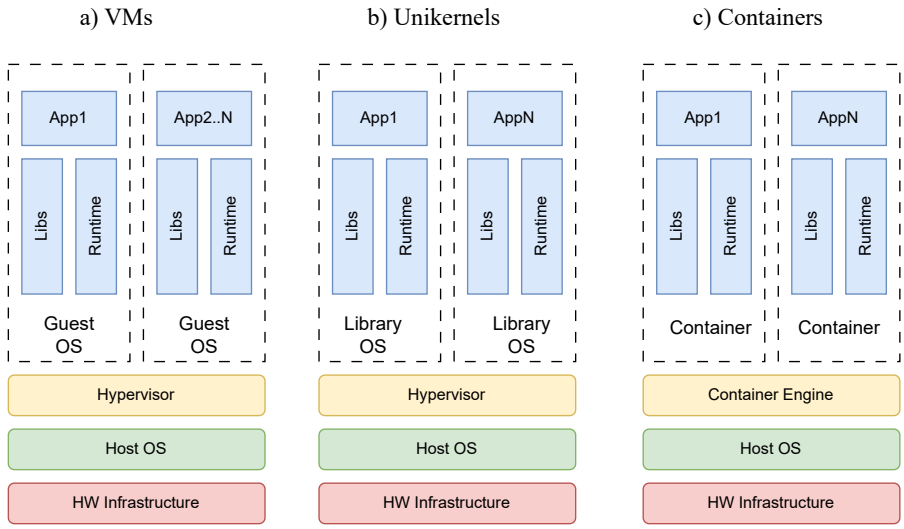


Figure 2.26: Architectures of virtualization technologies: Virtual Machines (a), Unikernels (b), and Containers (c)

network interfaces, and user IDs. Control groups (cgroups) allow fine-grained resource management by allocating specific amounts of CPU, memory, and I/O resources to each container [117].

- **Docker Containers**:

  Docker is another influential lightweight virtualization technology built upon containerization principles. It simplifies container management through a standardized format and tooling, enabling developers to package applications and their dependencies into portable containers. Docker containers are noted for their efficiency in resource utilization, rapid deployment, and portability across various environments, making Docker a preferred choice for microservices and network function virtualization [118].

  As shown in Fig.2.27 Docker operates based on a client-server architecture, where the Docker client interacts with the Docker daemon (Docker Engine), responsible for creating, running, and managing Docker containers. Containers are instantiated from images, which serve as read-only templates containing application code, runtime environments, libraries, and dependencies. Docker images are managed through registries, like Docker Hub, enabling easy distribution and version control [119].

  Docker Compose further extends Docker's capabilities by providing a streamlined method to manage multi-container applications. Through a simple YAML configuration file (*docker-compose.yml*), Docker Compose specifies how services, networks, and volumes are configured and interconnected. This enables users

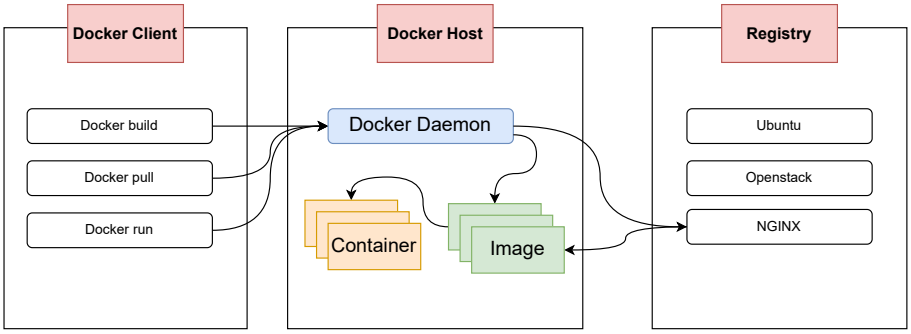


Figure 2.27: Architecture of Docker: Components (Client, Host, Daemon, etc.)

to orchestrate the startup, management, and scaling of complex, distributed applications comprising multiple Docker containers, enhancing both productivity and reproducibility across development, testing, and production environments.

Virtualization technologies, especially lightweight ones, e.g., based on containers as for the Docker solution [120], have become central to the flexibility and scalability of IoT networks. An example is the implementation of the edge computing paradigm, as shown in [121].

Docker has become a cornerstone of modern containerization, providing a lightweight and efficient solution for deploying and managing applications across different environments. It enables operating system-level virtualization, where multiple isolated containers share the same kernel, but each has its own file system, processes, and network interfaces. This isolation allows Docker to run applications independently without the need for a full-fledged virtual machine, reducing both overhead and resource consumption. Docker also simplifies the orchestration of these containers. With *Docker-Compose* [122], users can define and manage multicontainer applications in a single configuration file, making deployment and scaling straightforward.

Docker includes an engine that changes an image (stored somewhere) into a running container. By leveraging an overlay filesystem, additional layers can be added on the initial image, so that the final image consists of many overimposed layers, which are efficiently stored on disk and can be easily removed if no longer needed.

Furthermore, Docker enhances the portability of applications. Containers encapsulate all dependencies required to run an application, ensuring that the application can be deployed across different environments-whether on a local machine, a cloud server, or a remote device-without worrying about compatibility issues. This portability is particularly valuable in IoT networks, where devices may run on different hardware and software configurations yet need to interact in a standardized way.

Container-based virtualization achieves higher density deployment, reduced resource consumption, and lower startup latency, facilitating rapid deployment and efficient resource utilization essential for modern network hardware virtualization [116]. These characteristics make containers especially suitable for virtualizing network functions, such as Software-Defined Networking (SDN) and SDR components. Some research highlights the efficacy of lightweight virtualization in network scenarios, notably for SDN and Network Functions Virtualization (NFV), demonstrating substantial improvements in scalability, agility, and cost-effectiveness compared to traditional virtualization methods [123, 124]. Additionally, containerization significantly enhances the manageability and automation of network infrastructure, enabling rapid updates, dynamic scaling, and improved fault isolation [125].

Since this thesis focuses on the virtualization of EN, it is essential to first introduce relevant virtualization technologies. This section explores the foundational virtualization methods that are reshaping the landscape of network infrastructure.

Software-Defined Networking (SDN)

SDN represents a transformative shift in network architecture by decoupling the control plane from the data plane, which traditionally were tightly integrated within individual network devices. This separation allows centralized management through a software-based SDN controller, enabling greater agility, programmability, and flexibility in network configuration and operation. With a global view of the entire network provided by the controller, administrators can dynamically adjust network resources, enforce real-time policies, and optimize traffic routing efficiently based on changing network conditions and user requirements [126, 127].

The SDN architecture typically consists of three key layers. At the top, the application plane hosts network applications that specify policies and services such as Quality of Service Quality of Service (QoS), routing decisions, and security rules. These applications interact with the control plane through northbound interfaces, usually REST APIs. The control plane, containing the SDN controller, translates these application policies into actionable configurations and commands, relaying instructions to physical and virtual network elements. This communication occurs through southbound interface (SBI), most commonly OpenFlow, allowing the controller to program switches and routers dynamically. The bottom layer, the data plane, comprises the actual forwarding devices, including physical switches and routers, responsible for directing network traffic based on instructions received from the controller [126, 127].

The centralization of network control not only simplifies management but also enhances network security and facilitates seamless integration of cloud services, significantly improving resource allocation and operational efficiency in various scenarios, including cloud computing environments and industrial IoT setups [128]. Moreover, this centralized approach enables precise management of network performance parameters, such as bandwidth and latency, critical for supporting modern industrial applications and ensuring the high reliability and responsiveness demanded by contemporary network environments [129].

Network Functions Virtualization (NFV)

NFV has emerged as a transformative approach that fundamentally shifts how network services are provisioned and managed. Unlike traditional networks, where specialized hardware devices such as routers, firewalls, and load balancers are required, NFV allows these network functions to be implemented as software applications running on standard commercial-off-the-shelf (COTS) servers [130]. This shift leads to significant cost reductions, operational simplicity, and improved network flexibility, as resources can be dynamically allocated based on real-time demands without being constrained by specialized hardware dependencies [131, 132].

The core idea behind NFV involves abstracting network functions from dedicated hardware to virtual instances, known as Virtual Network Functions (VNFs). These VNFs are implemented as software running within VMs or containers hosted on general-purpose infrastructure, enhancing scalability and deployment speed. By using virtualization techniques, VNFs can easily be scaled, migrated, or duplicated across various physical servers, enabling rapid provisioning and adaptation of network services to changing requirements [133].

NFV also synergizes effectively with SDN, where SDN offers centralized control of network traffic flow, and NFV provides the flexibility in deploying network functions as software applications. The combination of SDN and NFV enables a highly programmable and dynamic network environment, capable of efficiently supporting complex scenarios like cloud services, 5G and beyond, and industrial IoT systems. Such synergy allows precise network resource allocation and real-time adjustments to network configurations, which are critical for supporting emerging applications requiring stringent QoS and security standards [126, 127].

Software-Defined Radio (SDR)

SDR represents a significant advancement in wireless communication systems, enabling greater flexibility, adaptability, and reconfigurability by implementing traditionally hardware-based radio functionalities through software. An SDR system shifts key radio functions, including signal modulation/demodulation, encoding, decoding, and signal processing from hardware components into software running on general-purpose processors or reconfigurable hardware such as Field-Programmable Gate Array (FPGA). This approach allows SDRs to dynamically adjust their operational parameters, such as frequency, modulation schemes, and bandwidth allocation, without hardware modifications, thus supporting multiple wireless standards and services through a single radio platform [134, 135, 136].

At the core of an SDR system is the separation between the radio frequency front-end hardware, which performs analog-digital conversions, and the software-based Digital Signal Processing (DSP) components. This architecture allows SDR devices to multiplex and demultiplex digitized In-phase and Quadrature (IQ) signal samples, facilitating the simultaneous operation of multiple virtual radio interfaces, each potentially running different wireless standards. For instance, SDR platforms can simultaneously support distinct communication technologies such as LTE, WiFi, and

Narrowband Internet of Things (NB-IoT) by dynamically multiplexing IQ signals into a unified data stream managed by a virtualization layer known as an SDR hypervisor [134, 135].

Furthermore, SDR's reconfigurability and virtualization capabilities make it an ideal candidate for integration into SDN and NFV paradigms, forming SDR. This integration creates a highly programmable and adaptive wireless environment, offering enhanced management capabilities, improved QoS, and greater scalability for various wireless applications, from industrial IoT to cellular communications and smart factories [137].

Figure 2.28: ADALM-PLUTO and its architecture

Table 2.7: Summary of ADALM-PLUTO features

Features	Details
Channels	RX1, TX1 (Full Duplex)
Frequency range	325 – 3800 (MHz)
Bandwidth Max.	20 MHz
Host Communication	USB 2.0
Digital Chip	FPGA (Xilinx Zynq Z-7010)
Controllable Parameters	Frequency, sample rate, bandwidth, gains
Additional Features	Portable, software reconfigurable

In our experiments within this thesis, the **ADALM-PLUTO** (see Fig.2.28) SDR device is utilized extensively due to its versatility, compactness, and affordability. The ADALM-PLUTO, also known as PlutoSDR, is an active learning module developed by Analog Devices, specifically designed for education, experimentation, and research in wireless communication [138]. It is built around the highly integrated Analog Devices AD9363 RF agile transceiver, covering a tunable frequency range from 325 MHz to 3.8 GHz. The device provides support for both Transmitter (Tx)

and Receiver (Rx) operations, whit an embedded a XilinxÂ® Zynq Z-7010 and FPGA logic to handle low-level radio tasks. PlutoSDR integrates essential components, including ADC/DAC converters, mixers, amplifiers, and filters, in a single programmable chip, enabling real-time digital signal processing and efficient prototyping of wireless systems. The ADALM-PLUTO device connects to a host computer via USB 2.0, allowing flexible interfacing with various SDR software frameworks, such as GNURadio, MATLAB/Simulink, and custom Python scripts. In our specific experimental scenarios, PlutoSDR has demonstrated its capability in rapidly prototyping communication protocols and implementing real-time SDR virtualization techniques. Its affordability and open-source support make it highly attractive for academic research and industry-oriented development projects alike [139]. The summary of the SDR features is given in Tab.2.7.

2.7 Literature Review: LoRaWAN

2.7.1 Introduction

This section reviews relevant literature on the application of SDR and lightweight virtualization technologies, particularly containerization, in LoRaWAN, wireless networks and IoT network architectures. The integration of these technologies enhances network flexibility, scalability, and manageability, presenting novel opportunities for abstracting and virtualizing communication functionalities.

2.7.2 SDR and Containerization for IoT Networks

The application of SDR in IoT networks has significantly expanded due to its flexibility in adapting to diverse communication standards. The work presented in [140] proposes a Cloud-Radio Access Network (C-RAN) architecture specifically tailored for LoRa networks, leveraging SDR for physical layer (PHY) cloudification. Functionalities are distributed into isolated Docker containers, demonstrating enhanced network manageability and scalability.

The feasibility of integrating SDR into IoT networks using containerization was further demonstrated by the work in [136], which utilized Docker containers across multiple SDR hardware platforms, highlighting significant benefits like ease of service management and reduced overhead compared to traditional setups. Similarly, Signal Monitoring, Analysis and Recording Tool (SMART) [141] employed Docker containers with SDRs on Network Attached Storage (NAS) units for efficient signal monitoring and data management, showcasing advanced applications including machine learning-based interference detection.

Moreover, Becker and Starobinski [142] introduced Snout, a middleware simplifying SDR application development by abstracting complex SDR interfaces, enabling broad applications from network security to IoT monitoring with minimal overhead.

The feasibility of employing SDR for PHY characterization is further validated

by [143], demonstrating cost-effective testbeds using commodity hardware and open-source software, particularly highlighting low complexity and affordability for Zigbee devices.

2.7.3 Virtualization Approaches in IoT and Wireless Networks

Lightweight virtualization technologies such as Docker and LXC have gained prominence due to their efficiency and flexibility. The integration of SDN and Docker in IoT architectures, as discussed by [144], significantly enhances network management capabilities by separating control and data planes.

Comparative performance analyses in [145] indicate that Docker containers operating on KVM-based virtual machines outperform traditional virtual machine setups in terms of CPU efficiency, memory usage, and network performance.

Security considerations and the expansive ecosystem of Docker have also been extensively examined [146, 147], highlighting both advantages and potential vulnerabilities associated with containerized deployments.

Further emphasizing network reconfigurability and improved resource utilization, [134] explores the adaptation of NFV and SDN to wireless environments, underscoring significant management and operational benefits.

2.7.4 LoRaWAN Node Virtualization

The virtualization of LoRaWAN end devices represents a transformative evolution, particularly through the use of container-based approaches. Recent studies have highlighted this capability, including [148], which introduces a container-based architecture utilizing the Constrained Application Protocol (CoAP) to integrate LoRaWAN nodes seamlessly with IP-based networks, addressing the limitations inherent in LoRaWAN's lack of native IP support.

A comprehensive approach toward full virtualization is preliminarily discussed in [149], where MATLAB-based models stream LoRa modulation samples through SDR. Further advancements reported in [150, 151] emphasize the benefits of containerizing node functionalities and backend services, significantly enhancing operational agility and scalability.

In particular, the study by [150] stresses how the containerization of both network node functionalities and backend services enables realistic LoRaWAN network emulation with reduced hardware requirements.

2.8 Conclusion: LoRaWAN

In summary, this section has provided a comprehensive overview of the foundational concepts and components of the LoRaWAN technology, emphasizing its sig-

nificance in enabling efficient long-range, low-power communications essential for IoT applications. The architecture of LoRaWAN, characterized by its star-of-stars topology, was detailed, highlighting the roles and interactions of ENs, gateways, and backend systems. Particular attention was given to the functionalities of the RN2483 end node module and the operation of gateways, including packet forwarding mechanisms. The backend's essential roles-NS, AS, and JS-were elucidated, showcasing their collective contributions to secure and reliable network operations.

Additionally, key aspects such as the LoRaWAN protocol stack, node classes (A, B, and C), the ADR mechanism, and joining procedures (OTAA and ABP) were thoroughly discussed. These elements collectively underline the flexibility and robustness of LoRaWAN networks in various operational environments, addressing critical challenges such as energy efficiency, scalability, and data integrity. Furthermore, the section examined advancements in the physical layer, particularly the Chirp Spread Spectrum modulation, emphasizing its role in extending communication range and improving resilience against interference.

This exploration establishes a solid foundational understanding, crucial for implementing and optimizing LoRaWAN-based solutions across diverse IoT contexts. The insights provided herein serve as the basis for subsequent discussions on virtualization technologies and their application in enhancing LoRaWAN deployments.

Chapter 3

Enhancing C-V2X Vehicle Platooning

3.1 Abstract

There is clear evidence indicating that the standardized SB-SPS in OoC possesses specific vulnerabilities. This unsatisfactory performance lies in the absence of coordination in channel scheduling and transmission time. Specifically, the determination to re-assign a resource after the reservation period expires is currently based on a random selection process, without consideration of the actual interference encountered by the utilized resources. In this study, a novel scheme is proposed to markedly improve the random selection method for Vehicular User Equipment (VUE) platooning use case.

3.2 Research Problem

Data losses and their causes [152] include *Interference* (simultaneous subchannel use by two VUEs), *Half-Duplex* (a VUE cannot transmit and receive simultaneously, leading to packet loss), *Propagation* (signal loss due to distance), and *Unsensed* (signals below the power threshold).

The Candidate CSR or a $E_{x,y}$ is randomly chosen from S_B. VUEs use a grant for multiple transmissions, managed by a RC set that randomly selects an integer value within the $[C_1, C_2]$ that depends on the RRI. The RC decreases with each transmission, and when it hits zero, VUEs may reselect a resource, influenced by the probability p_k within $[0, 0.8]$.

The issue of collisions arises when a particular resource is chosen, as nearby vehicles remain unaware of this selection until a signaling message is transmitted upon its initial use. Consequently, they might perceive it as available and consider it as a potential resource. This becomes particularly problematic when adjacent vehicles choose resources concurrently. To mitigate this, the mechanism employs a random number transmissions [153] $[C_1, C_2]$, aiming to space out their selections. However, this scheduling challenge remains inherent to the approach.

3.3 Theorized Discussion

The issue under consideration pertains to the temporal aspect during the selection phase, indicating that our attention should primarily be directed toward the time domain. As shown in Fig.3.1, the $n + T_1 \leq x \leq n + T_2$, then the value of x depends on two parameters, n and $[T_1, T_2]$. The variable T_1 denotes the duration needed by a VUE to identify potential resources and choose new ones (or the same) for transmission purposes. RRI defines the maximum allowed latency in milliseconds. The 3GPP standard does not fix a value of the RRI and its configuration is up to VUE implementation. The variable n is the instant at which the VUE decides on a new resource allocation that differs between the various VUEs within the network.

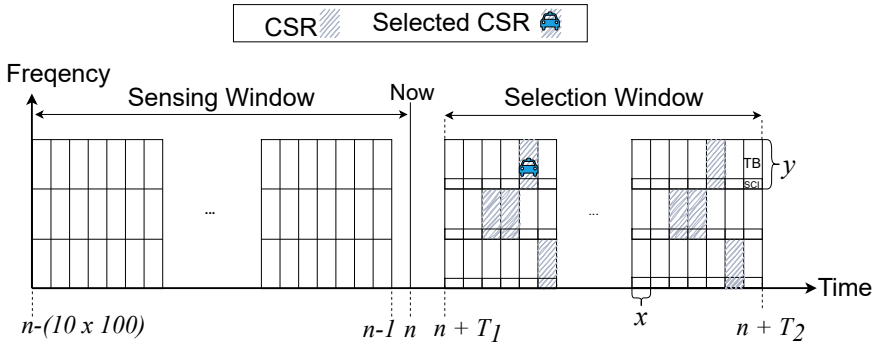


Figure 3.1: VUE autonomous sensing and resource selection.

3.4 Proposed Solution

The subsequent portion of this section outlines the strategies we have proposed to address the issue statement, specifically focusing on collision avoidance:

1. **Aligning Start Times for Optimal Coordination:** To enhance platooning vehicle communication, first, we need to ensure that multiple vehicles can simultaneously generate CAMs [154] to reach the common n where $[n + T_1, n + T_2]$ among the members of the platoon. This synchronization can be achieved through the use of beacon signals, a method that involves designated vehicles broadcasting beacon messages at regular intervals or Beacon Interval Time (BIT) [154], which contain precise time stamps derived from a highly accurate clock source, such as the Global Positioning System (GPS). The platoon leader beacon can act as a time server, providing a unified time reference for all receiving platoon members within the network. Upon receiving these beacon messages, each vehicle adjusts its internal clock to align with the time information provided. This effectively synchronizes the CAM generation cycle with that of the beacon vehicle and, by extension, with all other synchronized vehicles in the platoon network. This simultaneous action makes the selection windows of all platoon members the same as shown in Fig.3.2(b).

2. **Sorting S_B elements:** The S_B is sorted based on the x_i of each E_{x_i,y_i} where $i = 0, 1 \ldots, m-1$. The candidate single-subframe resources are arranged in ascending order based on the values x_i's. Without loss of generality, we can assume that $x_1 \leq x_2 \leq \ldots \leq x_m$. This ensures that the S_B list is organized in a way that makes it easier to perform subsequent operations efficiently, such as searching or selecting an $E_{x,y}$ based on its x.

3. **Random Subset instead of full Random Access:** Current MAC layer designs in OoC are made to be flexible enough to cater to various applications' needs. In both LTE-V2X and NR-V2X, the MAC layer uses a Random mechanism to

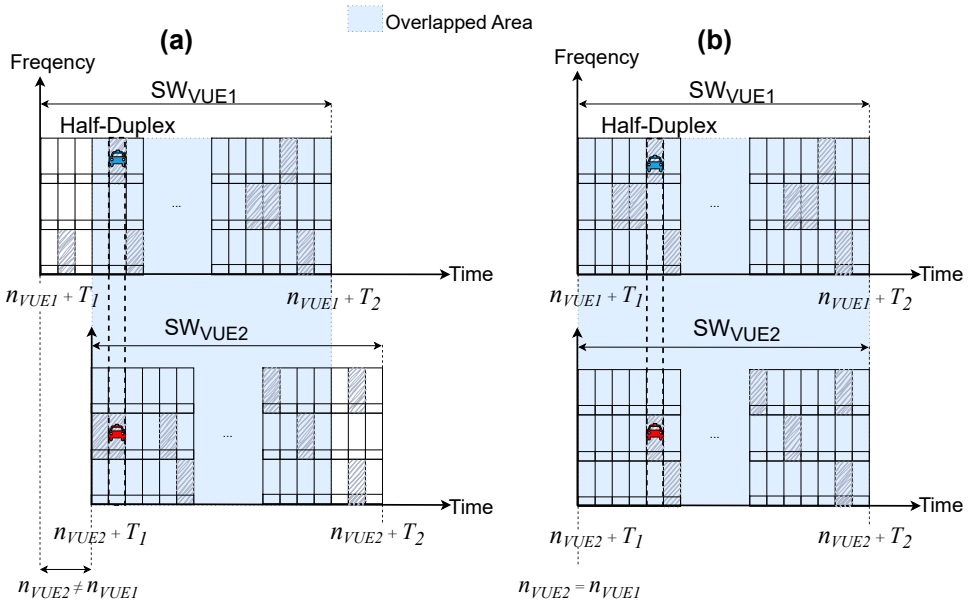


Figure 3.2: (a) Half-duplex collision between two vehicles with different selection times. (b) Half-duplex collision between two vehicles with simultaneous resource selection and overlapped Selection Windows (SW).

select the time and frequency resources for one transmission opportunity from the resources pool.

The Random Subset method is a variation of the traditional Random method, incorporating partitioning techniques to enhance the diversity of selected $E_{x,y}$s.

- The sorted set S_B is partitioned into subsets P, where P is the total number of members of the platoon. Each VUE is assigned one of these partitions.
- Unlike the traditional Random method, where the platoon members choose from the entire set S_B, each VUE in the Random Subset method picks an element randomly from its own partition.
- For VUEs not part of a platoon, the method falls back to the traditional Random method, ensuring that all VUEs have a method for selection.

Throughout the remainder of this chapter, the aforementioned three-step process will be referred to as the "*proposed*" method. It will be systematically compared and evaluated against the "*SB-SPS*" method to determine its efficacy and potential advantages. By comparing the probability of selecting distinct $E_{x,y}$ elements using the SB-SPS method versus the proposed method, in the SB-SPS method, each VUE has an equal chance of picking any element from S_B, with the probability of selecting distinct x values after the first VUE being influenced by the number (D) of

distinct x values and the uniformity of their distribution across S_B. The proposed method assigns VUEs to specific partitions of S_B, in order to minimize the likelihood of selecting the same x values, especially effective when D is large and x values are evenly distributed. The effectiveness of each method varies with the uniformity of x value distribution and the ratio of platoon members (P) to distinct x values, with the Partition method potentially offering higher distinct selection probabilities under conditions of large D and well-distributed x values.

3.5 Methodology

This section describes the implementation details of our proposed methodology, highlighting the key aspects of mobility management, communication protocols, and simulation parameters used for performance evaluation. Additionally, we will list the tools and technologies employed to implement and evaluate our approach, as shown in 3.3, which illustrates the tools used in this process.

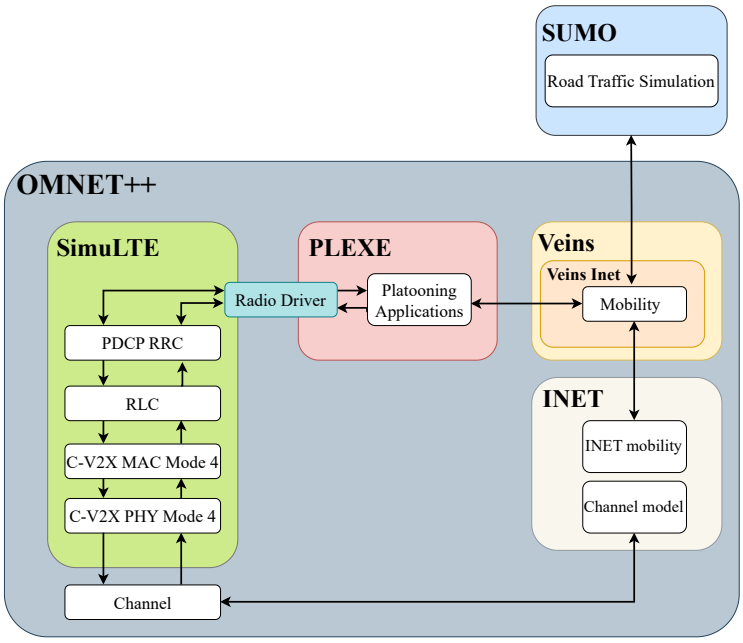


Figure 3.3: Architecture of the C-V2X simulation framework

In general, the network builds on the OMNeT++ (v5.6) simulator. We employ the SUMO (v1.18.0) simulator [155] integrated with PLEXE (v.3) [156], an extension of Veins (v5.0) [157], enabling the realistic analysis of the impact of the network on vehicles'behavior. In particular, it includes several CACC algorithms for platooning control. These algorithms exploit wireless communication and sensor measure-

ments to drive vehicles in a platoon configuration, maintaining small inter-vehicle distances without sacrificing safety. In this work, we consider one of these CACCs, namely the PLOEG [158] controller, which employs a time headway approach. For communication, the SimuLTE (v1.2.0) [159] and OpenCV2X (v1.4.1) [152] facilitate C-V2X through OoC mode, enhancing network simulation capabilities.

3.5.1 Wireless Network Simulation: OMNET++

For this project, OMNET++ [160], was chosen as the modeling environment. It is a C++-based discrete event simulator and, at the time of defining the model, was the most widely used for V2X applications. This popularity stemmed from the extensive use of ITS-G5 models such as Veins [161] and PLEXE [156], which were widely used to study this technology. OMNeT++'s modular nature and compatibility with PLEXE.

3.5.2 Network Component Modeling: INET

INET is the largest repository of networking models in OMNeT++, providing many of the essential network technologies needed to implement a full network. For example, in an experiment with a RSU, the veins module can be used for 802.11p-based communication between the RSU and the vehicle. If modeling RSU-to-Core Network performance, INET offers models for Ethernet, IPv6, TCP, and router functionality. Additionally, INET includes models for wireless propagation, channel modeling, and various WiFi variants [162].

In essence, INET serves as the primary resource for any network technology required for experimentation. Although it does not require integration with C-V2X, the main advantage of using OMNeT++ is its ease of integration into experiments. Users of the OpenCV2X model can use INET in their experiments to simulate more complex systems beyond direct V2X communication. This facilitates the exploration of more intricate use cases, such as edge experimentation, without significant implementation or integration overhead.

3.5.3 Vehicular Network Simulation: Veins

Veins is an open-source framework designed for simulating vehicular networks, built on top of two well-established simulators: OMNeT++ and SUMO. OMNeT++ handles the network simulation, while SUMO simulates road traffic and vehicle movement. The framework is specifically designed to model inter-vehicle communication (IVC) systems, allowing for the study of vehicular communication protocols in realistic traffic scenarios [157].

Veins is among the most established and widely adopted V2X simulation models, having been prevalent in research for over a decade. It provides robust integration with the traffic simulator SUMO via the TRACI API, and was primarily developed

for IEEE WAVE standard (802.11p) implementation. The key benefit of this integration lies in enabling a unified platform for direct and fair comparison between IEEE WAVE-based and C-V2X/NR-V2X technologies. Using the modular structure of OMNeT++, the same applications and scenarios can be seamlessly tested with both communication technologies without extensive reconfiguration. Furthermore, this integration facilitates exploring multi-Radio Access Network (multi-RAN) scenarios, where both technologies coexist on vehicles. These capabilities motivated the integration of OpenCV2X with Veins, thereby expanding the potential scope of investigation for the wider V2X research community.

The primary method of integration is through the TRACI API, which is implemented as part of Veins. This can be easily incorporated into any OMNeT++ model via the Veins Mobility interface, defined as a NED module. This interface facilitates seamless integration between OMNeT++ and SUMO, eliminating the need for significant additional implementation. By integrating the OpenCV2X model with Veins, this functionality is automatically inherited, requiring no extra user implementation. The only additional step is manually activating the TRACI server, which Veins uses during simulations, a process facilitated by a Python script.

3.5.4 Cellular Network Simulation: SimuLTE

OpenCV2X utilizes and extends SimuLTE as the foundation for its cellular network model. SimuLTE was originally designed to implement the cellular standard in OMNeT++ [163], specifically for eNodeB-enabled communication between the eNodeB and cellular devices. It provided a complete cellular stack, including radio and channel modeling, eNodeB scheduling, physical and medium access layers, as well as Packet Data Convergence Protocol (PDCP) [164] and RLC [165] layers. All stages of cellular communication were developed to be 3GPP standard-compliant. With the introduction of Device-to-Device (D2D) communication in 3GPP release 12 [166], this functionality was incorporated into the SimuLTE model, enabling cellular devices to transmit and receive data either directly or through an intermediary, without a direct connection to the eNodeB.

In the early development of OpenCV2X, SimuLTE had already created V2X scenarios that applied D2D-style communication, allowing vehicles to communicate with each other, scheduled by an eNodeB. Although initially limited in its V2X capabilities, this functionality provided a foundational base upon which OpenCV2X was built, allowing for further expansion into more complex V2X scenarios.

3.5.5 Sidelink Mode 4 Simulation: OpenCV2X

The OpenCV2X simulation model, developed by Brian McCarthy, provides an open-source, standards-compliant, and fully validated simulation environment for C-V2X and its successor, NR-V2X. This model is particularly tailored for analyzing scheduling behaviors and performance in vehicular networks employing C-V2X sidelink communication standards [69, 167].

In this study, we used OpenCV2X, which is a discrete event simulation model. It was the first full-stack open source model of the 3GPP C-V2X Rel. 14 model to be made available to the vehicular research community.

3.5.6 Application Layer: PLEXE

The PLEXE simulation model, developed by Michele Segata, is an advanced open-source extension of the Veins to support research and development in vehicular platooning [80]. This model realistically integrates vehicular communication, control algorithms, and detailed vehicle dynamics, making it an essential tool for validating cooperative platooning strategies before real-world deployment.

3.5.7 Microscopic Traffic Simulation: SUMO

A crucial aspect of V2X modelling is how vehicle mobility is simulated. For this purpose, OpenCV2X utilizes SUMO, an open-source road traffic simulation platform. SUMO is designed to model microscopic mobility for vehicles, pedestrians, and cyclists, though the focus here is primarily on vehicular mobility [168]. Developed by the German Aerospace Centre, SUMO has been extensively studied, and its underlying techniques, such as the Krauss car-following model, have proven effective in replicating real driver behavior [169]. As a result, SUMO has become the de facto standard for vehicular mobility simulation within the academic V2X community. All major simulation platforms integrate SUMO through its Traffic Control Interface (TRACI) API. Both Artery and Veins, which are described later, offer interfaces to SUMO via this API. Additionally, NS3 simulation models, such as ms-van3t [170], which provide similar services to Artery and OpenCV2X, also rely on SUMO, as does Matlab, which also integrates through the same API. Due to its open-source nature and ease of use for developing complex road networks, SUMO has been widely adopted by the broader simulation community. This capability is a powerful tool, not only for simulating wireless network interactions but also for studying the performance of the road network when V2X applications are integrated, as these can have a significant impact. Ultimately, SUMO is a core component of the OpenCV2X model, providing a robust road traffic simulation environment for our network simulations.

3.6 Configurations

Significant modifications were made to enhance platoon communication. A new beaconing class was introduced for periodic beacon transmissions within the platoon, ensuring synchronized timing across members.

Furthermore, the MAC layer in OpenCV2X was adjusted to implement a partition-based MAC strategy, promoting efficient selection mechanisms for vehicles within a platoon and fallback random selection for those outside platoons. Expanding on

this, Algorithm 1 initializes and sorts a vector of tuples, S_B, containing CSRs based on subframe values. Next, the algorithm aims to select an element from a set S_B in a way that minimizes the likelihood of multiple selectors choosing elements with the same x values. For platoon members, it divides S_B into partitions proportional to the number of platoon selectors, determining each selector's partition range.

Algorithm 1 Partition-based Selection Method

Input: $S_B, P, MACId, platoonMembers$

Sort S_B based on x values
if $MACId \in platoonMembers$ **then**
  $basePartitionSize \leftarrow \left\lfloor \frac{\text{Size of } S_B}{P} \right\rfloor$
  $offset \leftarrow$ Size of $S_B \bmod P$
  $extra \leftarrow 0$
  if $(MACId - \min(\text{platoonMembers})) < offset$ **then**
    $extra \leftarrow 1$
  end if
  $startIndex \leftarrow (MACId - \min(\text{platoonMembers})) \times basePartitionSize + \min((MACId - \min(\text{platoonMembers})), offset)$
  $endIndex \leftarrow startIndex + basePartitionSize + extra - 1$
  $index \leftarrow \text{intuniform}(startIndex, endIndex)$
else
  $index \leftarrow \text{intuniform}(0, \text{Size of } S_B - 1)$
end if
$selectedElement \leftarrow S_B[index]$
return $selectedElement$

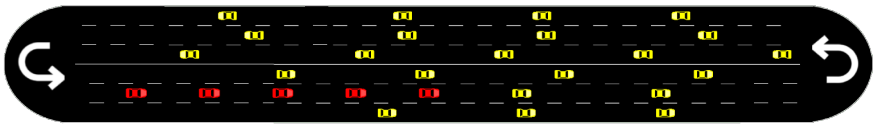


Figure 3.4: SUMO environment and the road topology used in simulation scenarios

Table.3.1 presents a summary of the simulation parameters, encompassing the radio configurations utilized for the C-V2X model. The absent configuration parameters were established in alignment with the default values of SimuLTE and OpenCV2X.

In Fig.3.4, a platoon of vehicles (red cars) travels at an average speed of 100 km/h, accompanied by background traffic (yellow cars) driven by humans, varying from 20 to 300 in different scenarios. The 3 km highway has four lanes. Within the platoon, the spacing between vehicles minimizes losses from weak signals, ensuring effective vehicle sensing.

Table 3.1: Simulation parameters

System	Parameter	Assumption
PLEXE	Leader's average speed	100 km/h
	Platoon size	{4, 5, ..., 19} Cars
	Simulation sampling rate	100 Hz
	PLOEG CACC time headway	0.5 s (15.89 m at 100 km/h)
OoC	PSCCH,PSSCH mode	Adjacent
	N_{subCH}	3
	SubCH size	16 RBs
	T_1	1 ms
	T_2	100 ms
	$[C_1, C_2]$	$[5, 15]$
	Prob. Resource Keep p_k	0.4
	BIT	100 ms
	Packet Size	200 byte
	UE Tx power	23 dBm
	Simulation time	1000 s

3.7 Results

Our analysis of packet loss focuses on the effects of *Half-Duplex* and *Interference*, emphasizing two key configurations: Platoon Size, where larger numbers of vehicles increase collision risk and reduce network throughput; and Background Traffic Size, where more nearby non-platoon vehicles can degrade network performance by increasing interference.

Running a 1000 second simulation scenario that includes a 5-vehicle platoon without background traffic, the heat map in Fig.3.5 reports a comprehensive comparison between two different methodologies, the traditional SB-SPS approach, and the newly proposed method. This graphical representation plots the relative occurrences of subframe selections (ranging from 1 to 100) across all vehicles within the platoon. Each cell represents a specific subframe, delineated along the x-axis, across various vehicles, listed on the y-axis. The color intensity of each cell correlates directly with the selection count, where lighter shades indicate higher frequencies of selection. This color gradient enables an immediate visual assessment of each method's performance, highlighting the most and least selected subframes by the vehicles. In the proposed method, it is observed that the selected CSRs exhibit greater distinctiveness, despite the fact that both methods randomly choose CSRs from those recommended by the PHY layer.

Fig.3.8 reports the inter-message delay distribution. Given a transmitter (leader) and receivers (platoon members) couple, there is a time (second) difference between the message that correctly arrived and the previous message correctly arrived. On the other hand, Update Delay (UD) reports on the delay the receiver experiences between updates from the transmitter. The scenario has a 5-vehicle platoon with-

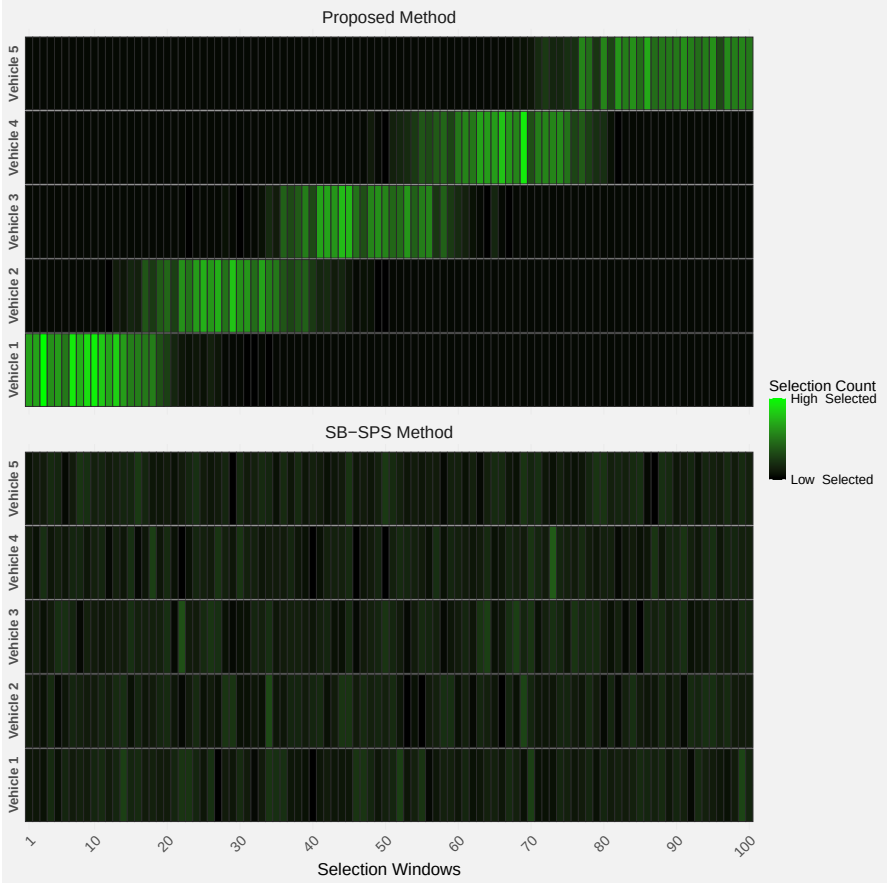


Figure 3.5: Distribution of Selection Count across Subframes in the selection window and both strategies functionality

out background traffic. The delay pattern is controlled by the consistent BIT = 100 ms intervals at which CAMs are planned to be sent. The scale used on both axes is logarithmic. It is crucial to note that the success of platooning is dependent on reliable and timely transmission and reception of information. Fig.3.7 illustrates that in the SB-SPS method, UD can be less than 100 ms but can vary significantly and introduce much higher delays above 500 ms. Such delays may result in vehicles not being aware of each other's dynamics, which increases the risk of collisions between them. In contrast, Fig.3.6 illustrates that the proposed method achieves a reduced UD, highlighting the enhanced reliability of this approach. This improvement comes from its ability to significantly decrease the incidence of *Half-Duplex* and *Interference* errors, which occur due to grant collisions.

Fig.3.9 compares the success rates of various numbers of vehicles in a platoon without background traffic. Success rates are calculated as the ratio of successfully

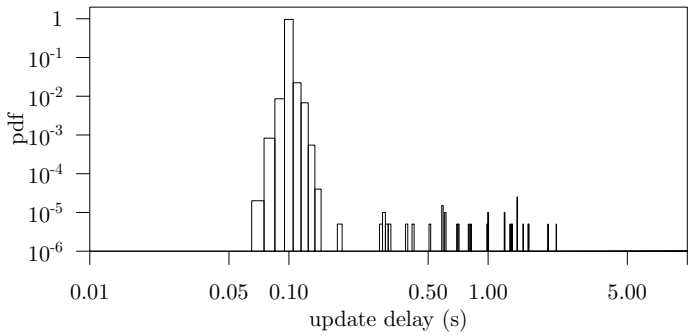


Figure 3.6: Proposed Method

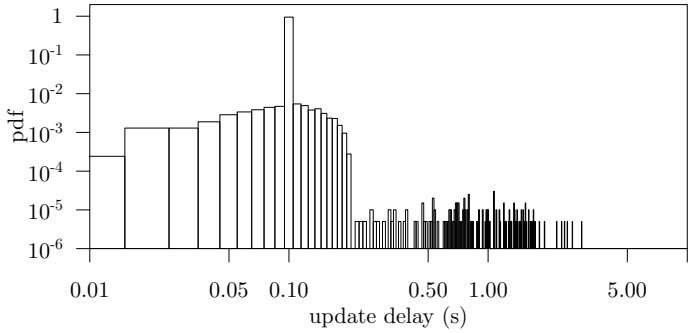


Figure 3.7: SB-SPS Method

Figure 3.8: Distribution of delays between two consecutive received messages

decoded messages (TBs) to the total expected messages, with the formula

$$\text{Success Rate} = \frac{\sum \text{Decoded Messages}}{(\text{Number of Vehicles} - 1) \times \sum \text{Sent Messages}}$$

The scattered points represent the average success rates of 20 repetitions, showcasing the variability and consistency of each method's performance. Lines indicate the trend of success rates modeled by linear regression, highlighting the general behavior of each method as the number of vehicles changes. The proposed method demonstrated a success rate improvement in the order of 1%. However, as confirmed by the higher slope of the linear fitting, the improvement decreases when the platoon size increases.

Furthermore, it aligns with the 3GPP requirements [171] for vehicle platooning, which require reliable V2V communications for up to 19 VUEs.

Fig. 3.10 evaluates the effectiveness of the methods, focusing on the success rates of TB decoding in scenarios with different numbers of background vehicles. In particular, it analyzes the performance of a four-vehicle platoon under simulated conditions.

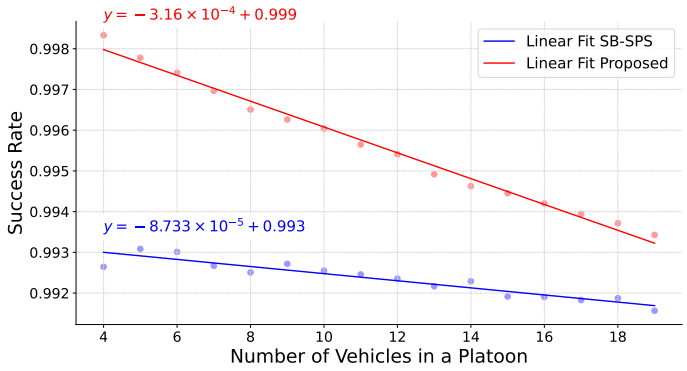


Figure 3.9: Comparing the performance of the methods as the number of platoon members increases.

The success rate is calculated using the following formula:

$$\text{Success Rate} = \frac{\sum \text{Successfully Decoded Messages}}{\text{Total Expected Messages}}$$

where the "Total expected messages" is the count of messages a platoon vehicle is expected to receive from the other three vehicles in the platoon, assuming ideal conditions with no message loss. The "Sum of successfully decoded messages" is determined by aggregating the number of TBs successfully decoded, focusing exclusively on communications between the platoon's vehicles. The performance of the methods was analyzed in scenarios with different numbers of background vehicles. These scenarios were repeated 20 times to calculate the average success rates, providing a comparative basis between the two methods. In addition, it features a line plot visualization that depicts the average success rate across different numbers of background vehicles for both methods, derived from polynomial regression models of the third degree. The findings indicate that, without altering the standard (SB-SPS), the rate of missed messages in a four-member platoon does not exceed 1% even with up to 300 background vehicles. However, the implementation of the proposed method significantly reduces this margin, approaching fewer missed messages and showcasing improved effectiveness with a higher initial success rate and less steep decline.

3.8 Conclusion and future outlook

The results of this study underscore the need for improved temporal coordination between VUEs in vehicle platooning. Improved synchronization can result in a network consisting of aware nodes, significantly reducing the number of blind transmissions. Our proposed solution contributes to this increased awareness, resulting in notable improvements in network performance through the reduction of UD.

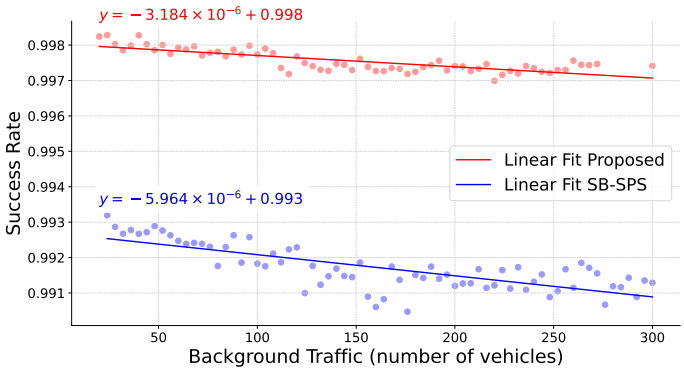


Figure 3.10: Comparing the performance of the methods as the number of background traffic increases.

Furthermore, this research highlights the critical need for continuous research and updates to the C-V2X standards to facilitate more effective vehicular communication, and the proposed method shows promise for improving the performance of LTE-V2X Mode 4, with potential applicability to NR-V2X Mode 2.

Chapter 4

Virtual LoRaWAN Node

4.1 First Phase: Implementing a software defined LoRaWAN node

4.1.1 Abstract

The use of microservices based on lightweight virtualization is nowadays a common practice for developing and deploying complex applications. Such an approach has been already profitably adopted for virtualizing functionalities of communication infrastructures. In this study, the design of a software defined LoRaWAN end node is discussed and the use of LXC is suggested to implement the decomposed functions as isolated microservices. A real-world test bench has been implemented and obtainable performance have been evaluated by means of purposely defined metrics. The feasibility of the proposed approach has been confirmed. In particular, results show that the main limitation is the throughput of the link connecting the host and the SDR device (in the order of 40MB/s).

4.1.2 Introduction

Many efforts have been carried out in the past to provide an effective abstraction of communication solutions, especially after the advent of the NFV, SDR and SDN concepts, that enabled full virtualization of all the components, down to the physical level. Due to the heterogeneity of SDR applications, a component-based model is mandatory in order to clearly define the border (i.e., the interface) and the functions (i.e., the semantic) of each communication mechanism. In this context, any application can be described as an ensemble of transformations applied during transmission and reception of the information of interest. Each transformation can be abstracted into a standalone entity that performs some kind of signal processing or control functionality. Such an approach allows for easier maintenance procedures and enables the reuse of already tested/developed blocks. Additionally, components can have multiple implementations, e.g., depending on the amount of available resources, sharing the same interface and realizing the same functionalities. In turn it means that some kind of virtualization is needed to develop the solution independently from the actual available hardware. It also means that some mechanisms to manage the deployment are required. Another important characteristic is the need for an underlying infrastructure for enabling the communication among components. Generally, a middleware software layer, often addressed as an hypervisor [135], is in charge of providing this infrastructure and a manager supervises the actually available hardware and software resources.

Several purposely designed frameworks have been proposed for addressing these needs, like those described in [172], but there is an increasing attention towards the adoption of traditional virtualization solutions. Notably, considering the need for high performance and real-time, LXC attracted interest as a viable solution for SDR/SDN applications [173]. In particular, this approach has been largely adopted for baseband unit (BBU) of wireless wide area networks, including mobile technolo-

gies (4G and 5G) and LPWANs, as in [174, 175]. In particular, such an approach has been implemented for BBUs in low-complexity solutions targeting IoT applications, like LoRaWAN [140]. In a previous work the authors have been the first to successfully adopt this approach to disaggregate and decompose functionalities of a LoRaWAN node, relying on a regular LoRa radio [151].

In this work the LoRaWAN node is fully virtualized, leveraging on a low-cost SDR instead of using a hardwired device. Real-world testbed has been arranged to experimentally evaluate obtainable performance when the node is connected to a regular backend. Docker has been considered as the reference LXC supervisor, exploiting the Compose tool to orchestrate the node functions virtualization. The diverse tasks are implemented by individual containers that are connected via a common MQTT-based databus, except for the component managing the SDR device, that exploit a TCP/IP client-server scheme across the USB physical link.

4.1.3 The LoRaWAN software defined node

According to previous description and focusing on the uplink data exchange, which is the most common traffic for a Class A LoRaWAN node during normal operation, the following different functionalities have been identified:

1. the end user application, that generates the (uplink) traffic;
2. the LoRaWAN payload generation, including the (uplink) traffic ciphering and insertion of the LoRaWAN header and trailers;
3. the LoRa encoding and baseband modulation of the LoRaWAN payload;
4. the conversion of the resulting digital stream into an analog signal and the mixing in the proper frequency band.

Steps from 1 to 3 are purely software services and are implemented as Docker-based LXCs, potentially hosted on different machines. Only the final step 4 leverages on the adoption of a SDR. A pictorial description of the proposed architecture, focused on the transmission of uplink messages, is shown in Fig.4.1. In order to highlight the heterogeneity permitted by LXC isolation, without losing generality, in the proposed solution:

- the “User App” container LXC1 for the step 1 embeds a simple application, written in JavaScript and developed using Node-RED, that periodically triggers the transmission of a message having a preset length;
- the “LoRaWAN” container LXC2 for the step 2 embeds a Python script taking care of properly formatting the LoRaWAN message (including ciphering and deciphering);
- the “LoRa” container LXC3 for the step 3 embeds a Matlab-based model of the LoRa radio, derived from the previous activity described in [176], transformed in a standalone application using the Matlab compiler;

- the "Serializer" container LXC4 for the step 4 embeds a streaming code developed in C, in charge of streaming the in-phase (I) and quadrature (Q) components.

A shared data bus must be provided for interconnecting the containers. As commonly happens, MQTT, leveraging on an additional Broker container, is considered, except for the LXC3-LXC4 link, utilizing a TCP/IP client-server approach.

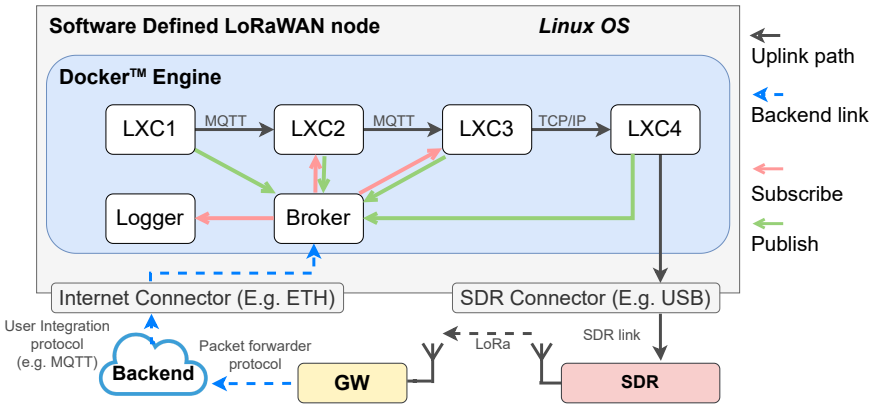


Figure 4.1: The architecture of the proposed software defined LoRaWAN node (uplink message transmission).

4.1.4 The Experimental Setup

Feasibility of the proposed approach has been verified in the test bench shown in Fig.4.2. The virtualized LoRaWAN device functionalities are executed in a PC, boasting an Intel i5-1135G7 CPU running at $2.40\,\mathrm{GHz}$, offering $16\,\mathrm{GB}$ of main memory, and $256\,\mathrm{GB}$ SATA SSD. Ubuntu (Version 20.04.6 LTS) is the host OS, running Docker (Version 26.0.1) and Compose (Version 1.28.5).The LoRa radio is implemented by the ADALM-PLUTO as explained in Sec. 2.6.6. It is a low-cost SDR operating up to $6\,\mathrm{GHz}$ with a $20\,\mathrm{MHz}$ bandwidth, that can be accessed via the Linux Industrial Input/Output (IIO) driver, which is part of the Linux kernel. It offers a USB 2.0 physical interface towards an host, which limits the throughput to some $\mathrm{MSa/s}$.

Referring to Fig.4.1, each time a new message has to be sent, LXC1 publishes the corresponding uplink topic; LXC2 is subscribed to such a topic and publishes in turn the LoRa encoded message; LXC3 is subscribed to such a topic and transfers the I/Q stream to the SDR via a TCP/IP client-server flow. Finally, the SDR has sample rate $FS = 2\,\mathrm{MSa/s}$, enough for the low-bandwidth of LoRa.

The test bench includes a LoRaWAN GW, the RG186 from Laird, running the Semtech UDP packet forwarder protocol. As regards the backend, *The Thing Stack* has been selected, due to its wide acceptance in both the industrial and academic domains; additionally, free usage is permitted for non-profit research.

Figure 4.2: The experimental setup including the LoRaWAN GW from Laird, the SDR, and the PC.

4.1.5 Experimental Results

Statistics of experiments are resumed in the Table 4.1. It can be highlighted that the most of the time required for transmitting an uplink message is spent to transfer the in-phase and quadrature sample streams from the serializer service LXC4 to the SDR. The $40\,\mathrm{MB}$-long sample buffer needs more than $1\,\mathrm{s}$ to be completely downloaded, a result in line with the expected throughput of the IIO driver[1].

Despite including the transfer delay, it is Interesting to note that the minimum values of Δ_4 and the Δ_7 statistics of messages transmitted using diverse payload actually differ by quantities comparable with the difference airtime duration.

Fig.4.3 and Fig.4.4 show the Δ_4 and the Δ_7 metric distributions using boxplots, clearly showing the impact of the different airtime duration. It must be remembered that incrementing the SF value approximately leads to double frame duration; deviations from expected values are mainly dictated by variability in the T_5 timestamp (also accounting for synchronization uncertainty).

4.1.6 Conclusion and future outlook

In this work, the use of lightweight virtualization for implementing a software defined LoRaWAN end node is discussed and analyzed. The device tasks are disaggregated and the stack levels are decoupled from the hardware using containers. The only layer requiring hardware is the physical one, implemented by means of a SDR. The setup of a real-world testbenche demonstrated the feasibility of the proposed approach. In particular, it has been possible to demonstrate the impact of link connecting the host system(s) running the containers and the SDR on the overall time performance.

[1]See for instance the manufacturer webpage (accessed on April 16,2024):
https://wiki.analog.com/university/tools/pluto/devs/performance

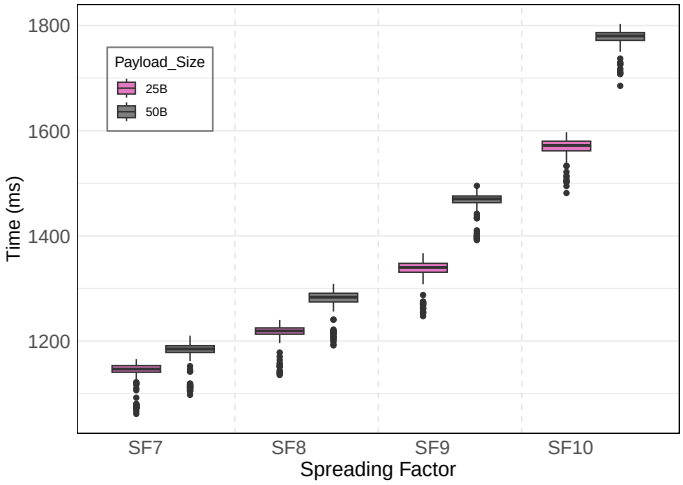


Figure 4.3: The Δ_4 metric distribution.

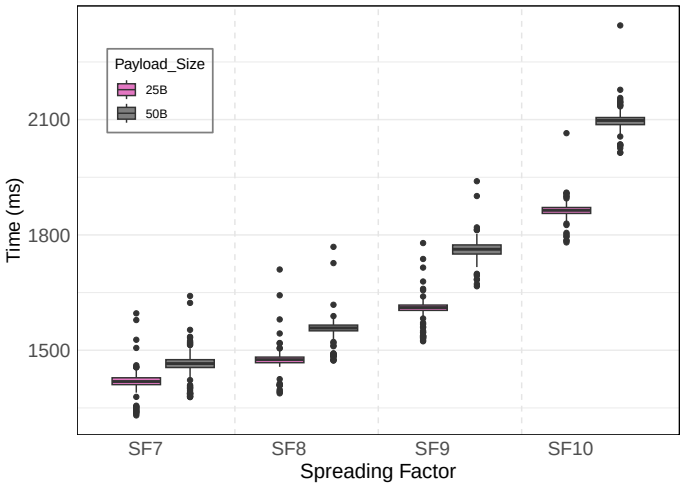


Figure 4.4: The Δ_7 metric distribution.

Table 4.1: Resume of Statistics

SF7-25B: Airtime T_{OA} = 82.2 ms							
Statistic (in ms)	Δ_1	Δ_2	Δ_3	Δ_4	Δ_5	Δ_6	Δ_7
Mean	0.16	45.06	4.45	1143.18	18.46	207.31	1418.62
Std	0.41	15.87	2.38	18.49	2.01	2.07	26.20
Median	0.13	43.68	4.10	1146.56	18.63	206.96	1418.69
Min	0.05	29.74	2.59	1061.36	0.08	204.26	1330.53
Max	7.02	188.42	33.28	1165.56	21.92	221.70	1596.02
5th Pctl.	0.06	33.16	2.80	1095.64	18.43	205.34	1390.45
95th Pctl.	0.19	56.02	7.08	1160.04	19.57	210.03	1443.06
SF7-50B: Airtime T_{OA} = 118.0 ms							
Statistic (in ms)	Δ_1	Δ_2	Δ_3	Δ_4	Δ_5	Δ_6	Δ_7
Mean	0.17	52.91	5.53	1179.89	18.68	207.33	1464.51
Std	0.07	17.10	2.53	21.51	0.95	2.97	29.45
Median	0.16	49.66	5.02	1184.84	18.61	206.87	1465.40
Min	0.06	35.71	3.37	1097.52	4.35	204.54	1378.12
Max	0.78	203.98	33.49	1210.23	22.25	247.94	1641.26
5th Pctl.	0.08	39.12	3.62	1114.24	18.43	205.37	1401.76
95th Pctl.	0.23	81.48	9.01	1200.17	19.64	210.35	1498.18
SF8-25B: Airtime T_{OA} = 143.9 ms							
Statistic (in ms)	Δ_1	Δ_2	Δ_3	Δ_4	Δ_5	Δ_6	Δ_7
Mean	0.13	43.23	5.84	1213.61	5.60	207.37	1475.78
Std	0.05	30.20	3.50	22.02	4.37	1.45	35.44
Median	0.13	35.78	4.95	1219.17	4.59	207.06	1475.64
Min	0.05	31.70	3.99	1135.28	0.11	204.78	1387.87
Max	0.57	303.65	33.45	1240.00	20.52	214.44	1710.09
5th Pctl.	0.07	32.93	4.24	1151.79	1.23	205.65	1409.98
95th Pctl.	0.18	56.76	10.77	1232.11	20.00	209.70	1505.20
SF8-50B: Airtime T_{OA} = 215.6 ms							
Statistic (in ms)	Δ_1	Δ_2	Δ_3	Δ_4	Δ_5	Δ_6	Δ_7
Mean	0.15	48.18	7.83	1278.32	13.65	207.51	1555.63
Std	0.05	16.31	2.05	22.30	8.48	2.22	27.83
Median	0.15	46.63	7.53	1283.23	18.82	207.06	1557.94
Min	0.06	37.01	5.71	1191.56	0.08	204.83	1473.09
Max	0.78	225.46	23.40	1308.69	68.56	224.64	1768.97
5th Pctl.	0.07	38.67	6.15	1216.63	2.03	205.51	1491.17
95th Pctl.	0.19	58.51	10.04	1298.63	20.54	209.51	1575.47
SF9-25B: Airtime T_{OA} = 267.3 ms							
Statistic (in ms)	Δ_1	Δ_2	Δ_3	Δ_4	Δ_5	Δ_6	Δ_7
Mean	0.13	47.77	9.46	1334.59	9.20	207.58	1608.74
Std	0.04	15.77	2.57	23.19	7.26	2.08	28.16
Median	0.13	43.71	8.92	1339.86	4.98	207.28	1611.10
Min	0.05	36.79	7.29	1247.21	0.34	204.78	1523.27
Max	0.40	180.11	35.93	1367.04	24.27	220.81	1778.89
5th Pctl.	0.07	38.61	7.65	1270.56	1.25	205.44	1546.20
95th Pctl.	0.16	61.21	11.94	1355.31	19.10	210.03	1632.20
SF9-50B: Airtime T_{OA} = 390.1 ms							
Statistic (in ms)	Δ_1	Δ_2	Δ_3	Δ_4	Δ_5	Δ_6	Δ_7
Mean	0.16	66.62	16.05	1465.49	6.68	207.30	1762.29
Std	0.04	19.67	3.81	20.45	6.28	1.40	29.86
Median	0.16	64.67	14.97	1470.34	4.57	207.10	1762.69
Min	0.06	44.84	11.44	1391.79	0.10	204.33	1666.49
Max	0.31	225.11	35.91	1495.20	19.82	216.49	1939.76
5th Pctl.	0.07	47.82	12.40	1404.46	0.48	205.58	1718.48
95th Pctl.	0.23	90.23	23.04	1484.27	18.88	209.37	1796.79
SF10-25B: Airtime T_{OA} = 493.6 ms							
Statistic (in ms)	Δ_1	Δ_2	Δ_3	Δ_4	Δ_5	Δ_6	Δ_7
Mean	0.12	58.68	18.39	1568.45	9.80	207.49	1862.93
Std	0.02	13.19	3.31	18.85	7.52	1.76	23.21
Median	0.12	57.15	17.48	1571.97	5.14	207.12	1864.00
Min	0.05	46.87	15.12	1481.41	0.01	204.64	1780.55
Max	0.21	208.23	46.24	1597.03	23.80	217.66	2064.72
5th Pctl.	0.06	49.42	16.16	1533.59	1.21	205.51	1837.78
95th Pctl.	0.16	69.47	23.04	1590.10	19.62	210.78	1884.47
SF10-50B: Airtime T_{OA} = 698.4 ms							
Statistic (in ms)	Δ_1	Δ_2	Δ_3	Δ_4	Δ_5	Δ_6	Δ_7
Mean	0.15	78.82	28.71	1775.86	6.58	207.43	2097.54
Std	0.03	21.20	5.91	18.63	5.83	2.42	29.35
Median	0.15	74.83	26.67	1780.18	4.62	207.11	2097.76
Min	0.06	61.59	24.69	1685.36	0.22	204.49	2013.88
Max	0.24	307.74	81.28	1803.13	18.73	231.17	2345.04
5th Pctl.	0.08	64.75	25.07	1729.21	0.99	205.48	2064.35
95th Pctl.	0.19	106.18	39.22	1793.42	17.64	209.84	2136.01

4.2 Second Phase: On the use of containers for LoRaWAN node virtualization

4.2.1 Abstract

This study investigates the virtualization of LoRaWAN end nodes through LXCs to improve scalability, flexibility, and resource management. By leveraging lightweight Docker-based virtualization, we break down the core functions of the LoRaWAN node, comprising the application, LoRaWAN, and LoRa layers, into modular containers. In this work, a fully virtualized end node is demonstrated. The obtainable performance is not only compared against the standard approach that leverages a LoRaWAN-compliant module but also against an emulated solution that mimics the desired functionalities purely in software. A controlled, uniform testbed, exploiting the capability of a virtual machine hypervisor to change the way the underlying hardware is abstracted to guest environments, is considered. Key metrics, including resource utilization and latency, are explicitly defined and evaluated. The results underscore the potential of container technologies to transform the deployment and management of communication solutions targeting IoT scenarios not only for the infrastructure but also for end devices, with implications for future advancements in wireless network virtualization.

4.2.2 Introduction

In recent years, wired networks have evolved to offer higher and higher flexibility, largely due to the shift from dedicated hardware solutions to software-driven functionalities. This trend has enhanced network performance and scalability, enabling network services to be provisioned, scaled, and optimized with remarkable efficiency. In the past, when dedicated hardware was used for implementing specific network functionalities, the update of an already deployed infrastructure required the introduction of new equipment. Supporting novel user application scenarios and/or protocols, which require disruptive network changes, resulted in a communication infrastructure redesign to comply with the new (standard) specifications. SDN and NFV technologies, just to mention a few, have played pivotal roles in this transformation by introducing programmability, flexibility, and cost-efficiency into the network infrastructure, for enabling the dynamic reconfiguration of available resources to meet diverse and changing demands [126].

Softwarization and virtualization of networks [129, 128] have recently extended into the wireless domain, when challenges such as the efficient use of spectrum and the need for low-latency, high-throughput communication are prominent [127]. The integration of the previously mentioned SDN and NFV solutions into wireless networks has opened new possibilities for flexible network and traffic management. Among the many advancements, the SDR concept has gained attention for its ability to virtualize radio functions, enabling real-time changes in radio access technology without requiring hardware modifications. In turn, these approaches permitted

the so-called *network disaggregation*, which can be split into (i) vertical disaggregation, when the software part is decoupled from the hardware part, allowing multiple combinations to be used, and (ii) horizontal disaggregation, when network functionalities are implemented using more granular elements interconnected by explicitly designed interfaces. Such an approach is a fundamental pillar of the latest mobile communication generation, minimizing both the capital expenditure (CAPEX) in deploying new architectures and the operating expenditure (OPEX) related to the upgrade of network functionalities, which in turn results in higher revenues for operators and service providers.

However, despite leveraging different business models, IoT-like applications also demand efficient and scalable network solutions. In particular, Low-Power Wide-Area Networks (LPWANs) emerged as viable solutions that trade off coverage and battery lifetime against data rate and latency. LoRaWAN, based on the LoRa physical layer, merges these requirements with simple infrastructure devices (the gateways) and an innovative backend, where most of the infrastructure complexity is concentrated. Such a combination makes LoRaWAN ideal for a range of IoT applications, as confirmed by its wide adoption in both academia and industry. However, as the LoRaWAN ecosystem evolves, there is a growing demand for more dynamic and reconfigurable network solutions to accommodate the diverse requirements of connected devices. Traditional LoRaWAN architectures, characterized by their monolithic and hardware-dependent setups, are ripe for innovation through the application of software-centric approaches. As regards connectivity to the backend, it must be highlighted that the gateway is a relatively dumb and transparent device that is only in charge of en/decapsulating the user payload in/from a LoRa frame; i.e., it is a simple hardware component dependent on the backend software. On the other hand, virtualizing the entire functionality of the LoRaWAN end nodes could offer a promising solution that allows modular, scalable, and flexible deployment of LoRaWAN applications.

By integrating concepts from the literature and our previous research [151, 149], this paper delves into the application of LXC [177] technology, specifically leveraging Docker lightweight virtualization, to disaggregate the LoRaWAN end node stacks into more manageable and dynamic components. Once LoRaWAN end-node functionalities are encapsulated within containers, the use of SDR allows for the complete virtualization of all the stack layers, significantly enhancing the deployment, scalability, and management of the overall application. However, despite the feasibility of this strategy having been somehow already demonstrated, a comprehensive characterization is still missing.

This work provides an answer to this research question. The novel and original contributions are as follows:

- The full virtualization of a LoRaWAN end node, across all the stack layers, is proposed based on the use of LXCs and a low-cost SDR device;
- A uniform environment for performance evaluation is described, leveraging virtualization of the underlying physical device to easily change memory and computational resource availability;

- Multiple scenarios, based on different implementations of the same network functionality, are described and realized in real-world experiments to carry out rigorous testing and measuring and comparing key performance indicators such as latency and system resource utilization;
- Analysis of the obtained results and insights about the benefits and challenges of the proposed architecture is carried out.

4.2.3 Research Gap and Contribution

Despite significant advancements, a comprehensive exploration of fully virtualized LoRaWAN end devices using LXCs remains limited. The existing literature primarily focuses on partial virtualization or Docker-based implementations. Addressing this gap, the present work evaluates resource requirements and scalability for complete LoRaWAN node virtualization using LXC technologies, comparing implementations based on MATLAB and C models. This evaluation considers computational, storage, and communication resources, providing detailed performance analyses under varying configurations, thus contributing significantly to the advancement of fully containerized LoRaWAN architectures.

4.2.4 The Proposed Approach

This study focuses on evaluating the performance of a regular end device against a fully decomposed functionalities node. Starting from Fig. 4.5, various disaggregation and decomposition strategies are detailed, showcasing how functionalities within end devices are segmented and containerized in different implementation scenarios. Each scenario uniquely handles the relevant stack layers, namely (a) the end-user application, (b) the LoRaWAN stack (including the management of the initial binding procedure and the payload (de)ciphering), and (c) the LoRa physical layer, utilizing either software virtualization or dedicated hardware. For the sake of clarity, a summary of the different scenarios is provided in Table 4.2; further details are presented in Chapter 2.6.

Table 4.2: Overview of the different experimental scenarios.

Scenario	Application Layer	LoRaWAN/DLL Layer	Physical Layer	Gateway	NodeDelay
Reference	Containerized	HW Module	HW Module	Real	+++
Fully Virtualized	Containerized	Containerized	Real	Real	+
Emulation	Containerized	Containerized	Absent	Virtual	++

Note: The symbols (+, ++, +++) represent qualitative assessments of the NodeDelay impact, with "+" indicating the largest (worst) delay, and "+++" indicating the shortest (best) delay values.

More specifically, based on the previous activities described in [151, 150, 149], the considered scenarios are as follows:

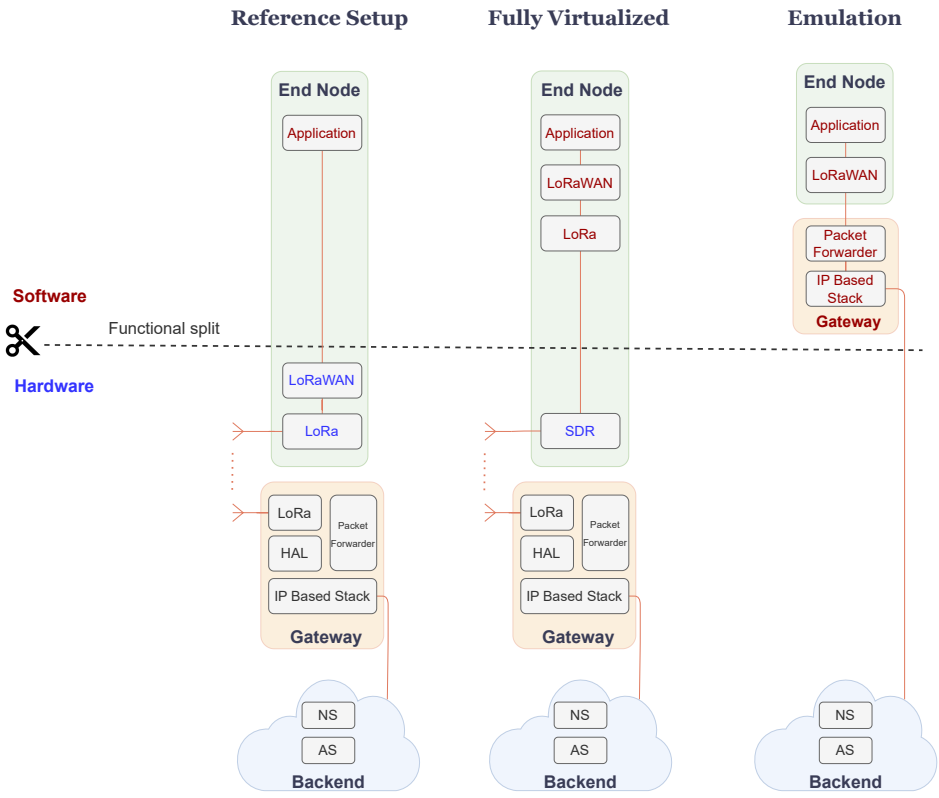


Figure 4.5: Overview of functional split scenarios in LoRaWAN node implementations.

- **Standard:** This is the reference scenario, representing a typical implementation of a LoRaWAN end node, leveraging a commercially available module. In this architecture, a clear separation between hardware-based communication functionalities and software-based application execution exists. Specifically, the LoRaWAN communication stack is fully handled by the hardware module, which also embeds the LoRa radio. Conversely, the end-user application, which is generally implemented by an external system/processor, is virtualized by means of an LXC.

- **Fully virtualized:** This scenario represents the proposed, holistic virtualization strategy. Disaggregation and decomposition of functionalities are implemented across all the layers of the communication stack. In particular, this setup relies on an LXC per layer and takes advantage of the versatility of SDR technology to manage the LoRa radio modulation and demodulation functionalities (occurring at the baseband) through software instead of hardware. Fortunately, despite LoRa being a proprietary modulation scheme patented by

Semtech, several reverse engineering (which is generally legal in the European Union if it is peformed for research and development purposes) efforts have been carried out in the past, e.g., as in [178, 179].

- **Emulation:** In this scenario, the end-node functionalities are still confined in LXCs, as in the fully virtualized case. However, the physical layer is completely abstracted out, demonstrating an implementation in which regular radio communications are omitted. Instead of utilizing LoRa radios to link the end nodes and the gateways, message forwarding towards the backend (and vice versa) is also containerized in a sort of virtual gateway.

As regards the latter scenario, its attractiveness as a (purely software) service for stress-testing real-world LoRaWAN infrastructure during the initial deployment or during normal operations must be underlined. Indeed, a variable, but controllable, load towards the backend can be easily injected, adding "emulated" devices, each one configured with a different update rate and user payload. Obviously, in this case, the impairments of the physical layer are not considered at all, since the LoRaWAN messages are directly conveyed to the backend using the virtual gateway.

4.2.5 Experimental Setup

An explicitly designed testbed has been developed to evaluate the computational, storage, and connectivity demands of the proposed approach. Efforts have been made to ensure that experiments are carried out in well-defined conditions, to easily and effectively compare results. In particular, we have decided to leverage a virtual machine to host the containerized functionalities. Accordingly, the underlying hardware components and platform are completely abstracted, and consistency is ensured across all the testing conditions. Indeed, despite the fact that we can suffer from additional latencies due to such an additional abstraction layer, in this work, we are mainly interested in comparing the impact of different levels of memory and computational resource availability, rather than analyzing absolute performance. For this reason, a uniform, configurable environment is a main concern.

The main characteristics of the host and guest systems are summarized in Table 4.3. As shown in Figure 4.6, the setup is based on an ML350 Gen10 server from HPE; the processor is an Intel Xeon Silver 4208 with eight cores, and the system memory is $64\,\mathrm{GB}$. It has to be stressed that the machine is not chosen according to the actual application requirements: as explained in the following, it is just a platform for running the VM instances to be compared under varying numbers of virtual CPUs and different amounts of available virtual memory. The host operating system is Ubuntu Server 20.04.6 LTS; as regards the hypervisor, the open-source Kernel-based Virtual Machine (KVM) has been considered, which is integrated in Linux from Kernel version 2.6.20. As a matter of fact, the KVM architecture mimics the Linux arrangement, which is based on (i) a native kernel module, switching the processor in a new guest state, executing the guest code, and leveraging the virtualization features offered by recent CPUs, and (ii) a virtual machine monitor, based on the Quick

EMUlator (QEMU), which virtualizes the hardware devices and provides some virtual networking functions. An interesting feature is that the guest code is emulated in a Posix thread, which can be managed by means of regular Linux tools; when a two-or four-core guest is needed, two or four threads are created, each of them calling the KVM kernel module to start the execution. If enough real cores are available (or through scheduling otherwise), concurrent execution is managed by the normal Linux scheduler, keeping the codebase minimal. The KVM environment is configured by means of the *Cockpit*, a free and open-source frontend tool, furnishing an intuitive web interface that allows us to manage and modify the relevant system features and resources easily and intuitively.

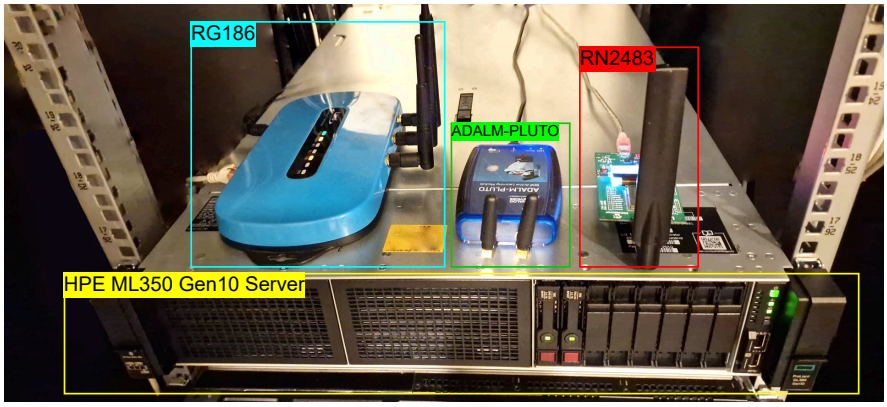


Figure 4.6: The experimental setup including the LoRaWAN GW, the SDR, the module, and the host.

In this way, the core setup remains consistent, but available (virtual) resources can be modified easily. Accordingly, different configurations can be devised to explore and evaluate significant aspects of the containerized LoRaWAN end device (see also Figure 4.8). The guest OS is Ubuntu 20.04.6 LTS; the VM is configured for using different numbers of virtual cores $vCPU \in \{1, 3, 5, 7\}$ and virtual memory space $vMemory \in \{4, 8, 16\}$GB; the virtual disk has a size of 215 GB.

As shown in Figure 4.6, the gateway used for communication is the Laird RG186 LoRaWAN gateway [180], which runs the Semtech UDP packet forwarder protocol, allowing communication between the end node and the backend services. In order to minimize impairments due to limited SDR transmitting power, noise, and interfering traffic, all the devices in the setup are positioned very close one each other. In terms of backend services, the system relies on *The Things Stack* (TTS) [181], a popular open-source LoRaWAN framework that provides all the necessary network management services.

Table 4.3: Hardware and platform specification of the experimental setup.

Host Processor	Intel(R) Xeon(R) Silver 4208 CPU @ 2.10 GHz (Cascadelake)
Host OS	Ubuntu Server 20.04.6 LTS
Guest vCPU Core (s)	{1, 3, 5, 7} Cores
Guest OS	Ubuntu 20.04.6 LTS
Guest vMemory	{4, 8, 16} GB
Guest vHardDisk	215 GB
Platform	Docker 27.4.0, Docker compose 2.20.0, MATLAB R2023b
Monitoring Tools	cAdvisor 0.39.3, node-exporter 1.8.2, Prometheus 3.1, Grafana 11.4

4.2.6 Configurations

The experimental framework adopted for this study utilizes four distinct setups, as shown in Figure 4.8. Indeed, other than the setups mimicking the aforementioned **Standard** and **Emulation** scenarios, two other setups have been defined for implementing different "flavors" of the **Fully Virtualized** scenario, as better detailed in the following. In all the experiments, the message length is $L_{PL} = 10\,\mathrm{B}$, and the message interval period is $T_{TX} = 30\,\mathrm{s}$. The overall experiment duration is $T_{EXP} = 3\,\mathrm{h}$. Other relevant settings are summarized in Table 4.4, resulting in the message duration of $T_{OA} = 61.7\,\mathrm{ms}$.

1. *Reference Setup*: A top-level container hosts an application developed using Node-RED [182], an open-source flow-based development tool, leveraging JavaScript code for data manipulation. This Node-RED flow (Fig.4.7) simulates the behavior of an end-user application that triggers the transmission of a fixed-length message based on a predefined schedule, possibly representing the uplink traffic of a sensor device in a real-world IoT-like application.

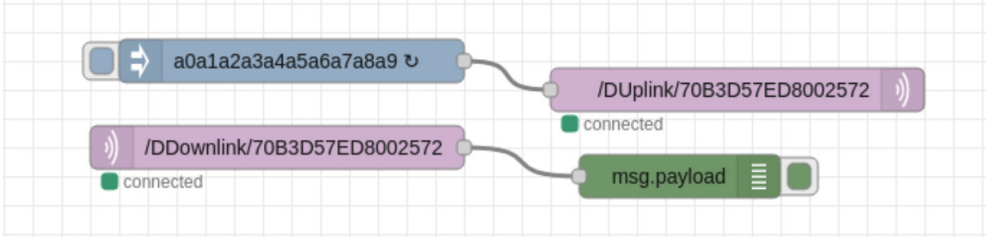


Figure 4.7: Node-RED flow showing MQTT communication.

A second container serves as the bridge between the containerized Node-RED application and the LoRaWAN module, the RN2483 from Microchip, shown in Figure 4.6, which is connected through a virtual COM above a USB link. This container handles the serialization of the application payload into the AT-commands that are interpreted by the module.

The containers communicate publishing and subscribing MQTT messages exchanged across a (containerized) Mosquitto broker [183].

2. *Fully Virtualized (MATLAB-based LoRa)*: In this setup, the application container implements the same functions as the standard setup. The user payload is processed by the next container, named "LoRaWAN". This container runs a C script responsible for managing various LoRaWAN operations; other than adding headers and trailers, it performs tasks such as ciphering and deciphering the payload and appending the necessary headers and trailers to prepare the data for transmission using the LoRa radio.

   Once the message is formatted, it is passed to the LoRa container, which hosts a MATLAB-based model of the LoRa modulation process. This model has been compiled into a standalone application and it is responsible for encoding (and decoding) symbols and performing the actual baseband (de)modulation. The MATLAB-based container outputs in-phase (I) and quadrature (Q) baseband samples, supposing a $Fs = 2\,\mathrm{MSa/s}$ sample rate. Each I/Q sample is a 2B-long signed integer value.

   The subsequent container, named Serializer, is responsible for streaming the I/Q components to the SDR device for actual signal transmission. The low-cost ADALM-PLUTO [139] from Analog Devices is adopted, capable of operating up to $6\,\mathrm{GHz}$ with a maximum real-time bandwidth of $20\,\mathrm{MHz}$ imposed by the maximum sampling frequency of $Fs_{MAX} = 61.44\,\mathrm{MSa/s}$. It is a cost-effective experimentation platform that leverages the Analog Devices AD9363 transceiver and the Xilinx Zynq Z-7010 System On Chip (hosting both an FPGA for pre-processing and an ARM dual-core CPU for supervision) capable of streaming the I/Q samples from the ADC and towards the DAC, with a 12-bit vertical resolution. This SDR is supported by the Linux Industrial Input/Output (IIO) driver, which is part of the Linux kernel. The ADALM-PLUTO is connected to the system via a USB 2.0 interface, which imposes a throughput limit of several MSa/s, adequate for the low-bandwidth requirements of LoRa. As also described in [149], a fixed buffer of $10\,\mathrm{MSa}$ stores the samples (i.e., $40\,\mathrm{MB}$ are transferred each time to the SDR), enough to contain the actual message. Indeed, since the actual sample rate is limited to $Fs = 2\,\mathrm{MSa/s}$ and $T_{OA} = 61.7\,\mathrm{ms}$, the number of samples per message is in the order of $N_{Sa} = FS \cdot T_{OA} = 123.4\,\mathrm{kSa}$.

   All containers communicate via an MQTT broker, except for the Serializer, which sends the I/Q streams to the SDR using a TCP/IP client–server flow.

3. *Fully Virtualized (C-based LoRa)*: In this setup, our proposed method adopts a virtualization strategy similar to that of the MATLAB-based SDR driven scenario, with a distinct implementation approach for the LoRa layer. In this case, the LoRa model is developed using the C programming language, known for its efficiency and control over system resources, and it is executed by its own dedicated container.

4. *Emulation*: This setup demonstrates the flexibility of the proposed approach. As already described in Section 4.2.4, this configuration does not involve any

actual LoRa transmissions. Instead, communication is emulated through a virtualized component known as the Virtual Gateway (VGW). The VGW collects the Application container output, which is immediately forwarded to the backend by means of the implemented packet forwarder functionality.

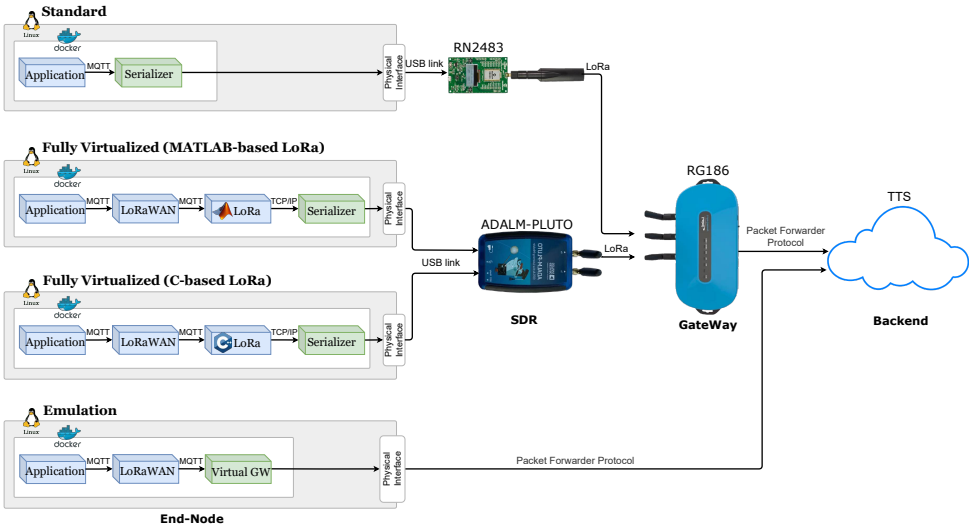


Figure 4.8: Experimental setups for virtualized LoRaWAN environments, showcasing different communication and network configurations across scenarios.

Table 4.5 shows the network traffic for each container in the different experimental scenarios, with values for incoming (I) and outgoing (O) traffic, obtained by means of the *docker stats –no-stream* command. These values represent the amount of data the container has sent and received over its network interface, calculated as the incremental changes in the actual Net I and Net O metrics over time. This information provides a clear picture of data movement in kilobytes (kB) and megabytes (MB), highlighting the network load and communication patterns within the virtualized environments used in this study. These statistics are consistent with the expected traffic previously described. As expected, there is no difference in the Application traffic in all cases; similarly, both the MATLAB- and the C-based LoRa containers for the Fully Virtualized configurations provide the same traffic levels.

4.2.7 Results—Resource-Related Metrics

This section presents and discusses the results of the experiments carried out using the previously introduced Fully Virtualized configurations; the focus is on resource-related metrics.

To compare the resource usage across containers, we employed a monitoring stack comprising **Prometheus** and **Grafana** [184], well-established tools for collect-

Table 4.4: Configuration parameters for LoRaWAN end-node setup.

Stack	Parameter	Assumption
Application	Payload Size	$L_{PL} = 10\,\mathrm{B}$
	Message Interval	$T_{TX} = 30\,\mathrm{s}$
	Experimental Period	$T_{EXP} = 3\,\mathrm{h}$
LoRaWAN	Version	LoRaWAN Specification 1.0.4
	Class	A
LoRa	Spreading Factor	$SF = SF7$
	Coding Rate	$CR = 4/5$
	Band	$868\,\mathrm{MHz}$
	Bandwidth	$B_C = 125\,\mathrm{kHz}$
	Airtime	$T_{OA} = 61.7\,\mathrm{ms}$

Table 4.5: Network traffic.

Scenarios	Container	Δ Network Traffic	
		I	O
Standard	Application	0	0.1 kB
	Serializer	0.4 kB	1.2 kB
Fully Virtualized (M-LoRa)	Application	0	0.1 kB
	LoRaWAN	0.4 kB	0.6 kB
	LoRa	1.7 kB	0.5 MB
	Serializer	1.3 MB	40 MB
Fully Virtualized (C-LoRa)	Application	0	0.1 kB
	LoRaWAN	0.4 kB	0.6 kB
	LoRa	1.7 kB	0.5 MB
	Serializer	1.3 MB	40 MB
Emulation	Application	0	0.1 kB
	LoRaWAN	0.4 kB	0.6 kB
	Virtual GW	0.6 kB	1.7 kB

ing and visualizing time series. This setup provides detailed insights into the resource consumption of different system components in real time. For container-level resource monitoring, we used **cAdvisor** (Container Advisor) [185] to expose container-specific metrics, while **node-exporter** [186] offered system-level metrics.

In addition to the monitoring stack, we also used the *docker stats –no-stream* command-line tool, which provides real-time performance data for running containers. This tool was particularly useful for tracking the Process IDs (PIDs) and Network Traffic I/O for each container.

Resource Usage Comparison: MATLAB- vs. C-Based LoRa

Both the C-based and MATLAB-based `LoRa` containers are the most demanding, since algorithms for the baseband (de)modulation process must be executed. Figure 4.9 illustrates the CPU usage of the `LoRa` containers as a percentage of the total computational capacity of the host machine. To monitor the container's CPU usage, cAdvisor was employed to expose container-level metrics, `container_cpu_usage_seconds_total`, while node-exporter provided system-level metrics, `machine_cpu_cores`. These metrics were scraped by Prometheus every $15\,\mathrm{s}$ and queried using the following **PromQL** expression:

```
sum(rate(container_cpu_usage_seconds_total
{container_label_com_docker_compose_service="LoRa"}[5m]))
  /
avg(machine_cpu_cores)
* 100
```

The numerator (`rate(container_cpu_usage_seconds_total[5m])`) calculates the rate of change in CPU time consumed by the `LoRa` containers, averaged over a 5 min long window. The denominator (`avg(machine_cpu_cores)`) represents the total number of vCPU cores available on the host machine; the result is multiplied by 100 to express the CPU usage as a percentage.

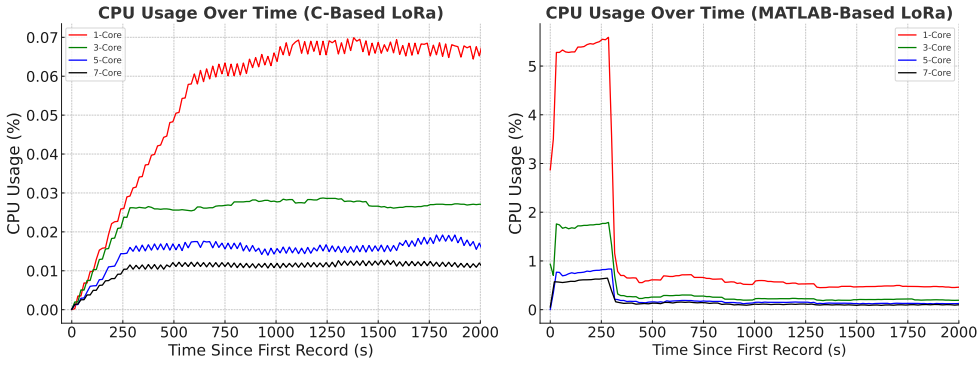


Figure 4.9: CPU usage of the C- and MATLAB-based LoRa containers as a percentage of the total capacity.

It is interesting to highlight the relevant overhead introduced by the MATLAB runtime environment, which, differently from the C-based implementation, exhibits a huge vCPU usage in the startup phase and is an order of magnitude more demanding than the plain C counterpart. As regards the number of the vCPUs, a relevant impact moving from one to three vCPUs (the usage is almost divided by a factor of two) can be shown, which is less evident when moving from five to seven vCPUs.

Table 4.6 presents the distribution of PIDs between different vCPU cores for each

container in the various configurations. A significant observation is the contrast in PID utilization within the LoRa containers in different scenarios. In the Fully Virtualized (C-LoRa) scenario, where the LoRa container is implemented using C, the container, as expected, consistently uses only a single process. In contrast, in the Fully Virtualized (M-LoRa) configuration, where the LoRa container is developed using MATLAB, the number of processes involved is substantially higher and it increases as more vCPU cores are allocated to the system. This behavior reflects the usage of the MATLAB native signal processing functions for implementing the algorithms of interest.

Figure 4.10 illustrates the memory usage of the LoRa containers, represented as a percentage of the total memory available on the guest machine (changing in the set {4,8,16}GB), when the number of available vCPU cores is limited to one core. These metrics were scraped by Prometheus every $15\,\mathrm{s}$ and queried using the following PromQL query:

```
sum(rate(container_memory_usage_bytes
{container_label_com_docker_compose_service="LoRa"}[5m]))
  /
avg(node_memory_MemTotal_bytes)
* 100
```

As expected, memory usage scales almost linearly with the overall amount. Also, in this case, the C-based LoRa container generally uses less memory and shows more stable usage compared to the MATLAB-based one.

Table 4.6: Distribution of PIDs across CPU cores for containers in different experimental scenarios.

Scenarios	**Container**	**CPU Cores**			
		# 1	**# 3**	**# 5**	**# 7**
Standard	Application	21	21	21	21
	Serializer	1	1	1	1
Emulation	Application	21	21	21	21
	LoRaWAN	3	3	3	3
	Virtual GW	5	5	5	5
Fully Virtualized (C-LoRa)	Application	21	21	21	21
	LoRaWAN	3	3	3	3
	LoRa	**1**	**1**	**1**	**1**
	Serializer	5	5	5	5
Fully Virtualized (M-LoRa)	Application	21	21	21	21
	LoRaWAN	3	3	3	3
	LoRa	**25**	**31**	**36**	**39**
	Serializer	5	5	5	5

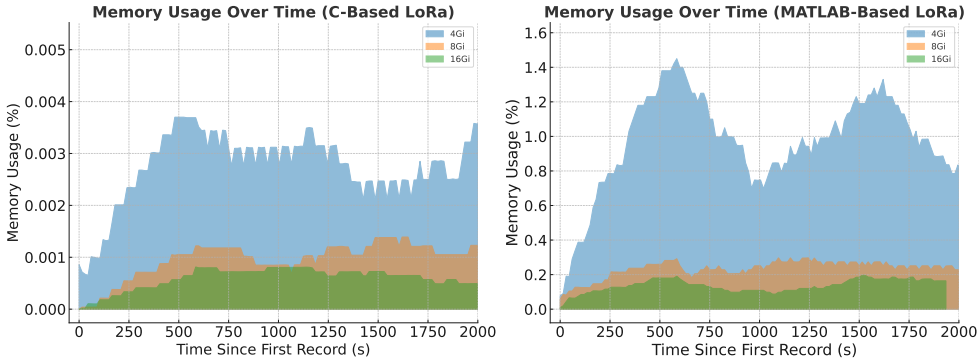


Figure 4.10: Memory usage of the LoRa containers as a percentage of total guest memory capacity.

As a concluding remark, it is possible to state that these stress the importance of an effective implementation and allow us to figure out the feasibility of this fully virtualized LoRaWAN node on an embedded platform.

4.2.8 Results—Time-Related Metrics

After highlighting the need for resources, in this section, the results of the experiments carried out using the previously introduced configurations are presented and discussed, focusing on time-related metrics.

Time Synchronization

Given the distributed nature of the system, both the local devices and the backend need to be synchronized to a common reference time, such as UTC, to assess time-related performance. The LoRaWAN backend is inherently synchronized with UTC. Therefore, the local PC, which hosts the containers, runs the Network Time Protocol (NTP) service [187], managed by *chrony* [188] , to periodically adjust the local time, based on measurements of offset and delay from previous steps.

The synchronization uncertainty is calculated using the formula $u_{sn} = \sqrt{\mu_{sn}^2 + \sigma_{sn}^2}$, where μ_{sn} is the average uncompensated systematic error introduced by the operating system (i.e., the average offset), and σ_{sn} represents the random error (i.e., the standard deviation of the offset). The system logs were collected over a three-hour period to measure the residual time offset after compensation, yielding $u_{sn} \approx 3.47\,\mathrm{ms}$.

It is important to note that when calculating statistics based on differences in timestamps from the same clock, the systematic errors tend to cancel out. As a result, $u_{sn} = \sigma_{sn} \approx 0.5\,\mathrm{ms}$. A similar level of uncertainty can be attributed to timestamping accuracy, as discussed in prior studies [189]. It is also expected that the synchroniza-

tion and timestamping accuracy of the backend servers do not exceed these values.

Time-Related Metrics

To evaluate the performance of each scenario from the time domain point of view, we tracked the uplink message path by capturing a series of timestamps at each stage. Within the virtualized environment, each container recorded the time at which a message entered and exited. The relevant figures of merit for direct comparison were obtained by computing the time taken at each step for the different configurations. An additional container, named Logger and specifically tasked with this function, managed the publication of these timestamps on the MQTT database. In the backend sections, timestamp collection was facilitated by the TTS platform, which automatically timestamps the reception of the uplink message at the GW, NS, and AS. Therefore, other figures of merit can be evaluated for the characterization of the complete path from the field to the end user, e.g., for direct comparison with other setups discussed in the literature.

Figure 4.11 illustrates the duration of each process in the different experimental configurations, identified by the subscript $Conf \in \{S, FM, FC, E\}$ for Standard, Fully Virtualized (M-LoRa), Fully Virtualized (C-LoRa), and Emulation, respectively. The timestamps T_{Confi}, where the subscript i is nothing but the progressive index, as shown in Figure 4.11, permit the computation of duration intervals D_{Confi}, better detailed in the following for the different configurations. In particular, in all configurations, the **Node Delay** D_{ConfND}, representing the time spent within the virtualized environment, and the **End-to-End Delay** D_{ConfEE}, representing the overall time needed to complete a roundtrip LoRaWAN transaction, are computed as well.

Standard Configuration:

- $D_{S1} = T_{S2} - T_{S1}$, representing the processing time taken by the Serializer service;
  - T_{S1} marks the moment when the Serializer service first receives a request for an uplink communication.
  - T_{S2} corresponds to the instant when the Serializer service has completed processing the request and forwards the uplink payload to the RN2483 module.
- $D_{S2} = T_{S3} - T_{S2}$, representing the time taken for the message to be gathered by the gateway;
  - T_{S3} is defined as the instant when the physical gateway (GW) acknowledges the reception of the new uplink communication originating from the module.
- $D_{S3} = T_{S4} - T_{S3}$, representing the time taken for the message to travel from the gateway to the NS;
  - T_{S4} identifies the moment the NS is notified of the new uplink communication.

- $D_{S4} = T_{S5} - T_{S4}$, representing the time taken for the message to travel from the NS to the AS;
  - T_{S5} identifies the moment the AS is notified of the new uplink communication.
- $D_{SEE} = T_{S5} - T_{S1}$ is the End-to-End delay;
- $D_{SND} = T_{S2} - T_{S1}$ is the Node delay;

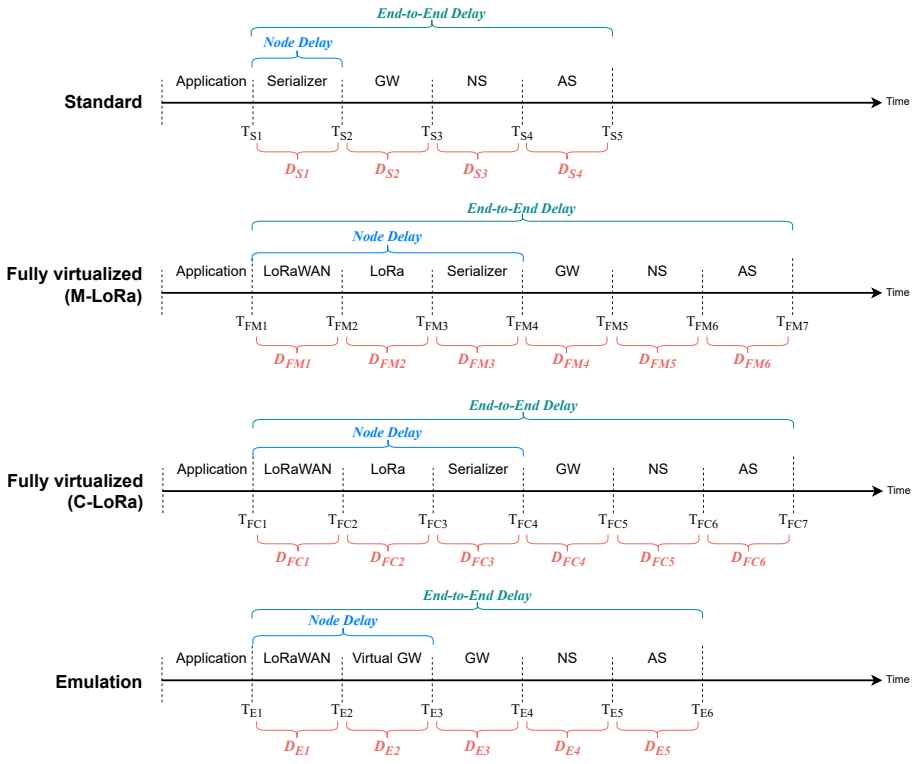


Figure 4.11: Time-related metrics for each stage of the uplink message path across different experimental configurations.

Fully Virtualized (M-LoRa) Configuration:

- $D_{FM1} = T_{FM2} - T_{FM1}$, representing the time spent within the containerized service for managing the LoRaWAN layer, from receiving the uplink request to relaying the processed payload to the next layer.
  - T_{FM1} marks the moment when the LoRaWAN service initially receives a request for a new uplink communication.

  - T_{FM2} corresponds to the instant when the LoRaWAN service has processed the request and relays the uplink payload to the subsequent lower layer.
- $D_{FM2} = T_{FM3} - T_{FM2}$, representing the time spent in the containerized service for managing the LoRa layer, from receiving the payload to generating the I/Q samples for transmission.
  - T_{FM3} is the instant when the LoRa service has completed the LoRa encoding and baseband modulation, meaning the actual I/Q discrete samples have been generated.
- $D_{FM3} = T_{FM4} - T_{FM3}$, representing the time spent in the containerized service for setting up the link between the **Serializer** service and the SDR, ensuring the system is ready to transmit the samples.
  - T_{FM4} denotes the point at which the Serializer service has successfully established the connection with the SDR and is ready to stream the samples.
- $D_{FM4} = T_{FM5} - T_{FM4}$, representing the time spent in the containerized service for streaming the I/Q samples to the SDR and propagating the uplink signal through the air until it is received by the gateway.
  - T_{FM5} is defined as the instant when the physical GW acknowledges the reception of the new uplink communication originating from the SDR.
- $D_{FM5} = T_{FM6} - T_{FM5}$, representing the additional time required for the uplink communication to reach the NS, capturing the time taken for the message to travel from the gateway to the NS.
  - T_{FM6} identifies the moment the NS is notified of the new uplink communication.
- $D_{FM6} = T_{FM7} - T_{FM6}$, representing the additional time required for the uplink communication to travel from the NS to the AS.
  - T_{FM7} identifies the moment the AS is notified of the new uplink communication.
- $D_{FMEE} = T_{FM7} - T_{S1}$ is the End-to-End delay;
- $D_{FMND} = T_{FM4} - T_{S1}$ is the Node delay;

Fully Virtualized (C-LoRa) Configuration:

Durations, D_{FCi}, D_{FCEE}, D_{FCND}, and time points, T_i, in this configuration are analogous to those in the previous one, the Fully Virtualized (M-LoRa) configuration, with the only difference being their naming convention.

Emulation Configuration:

- $D_{E1} = T_{E2} - T_{E1}$, representing the time spent within the containerized service for managing the LoRaWAN layer, from receiving the uplink request to relaying the processed payload to the next layer.

  - T_{E1} marks the moment when the LoRaWAN service initially receives a request for a new uplink communication.
  - T_{E2} corresponds to the instant when the LoRaWAN service has processed the request and relays the uplink payload to the subsequent lower layer.
- $D_{E2} = T_{E3} - T_{E2}$, representing the time spent within the VGW container, from receiving the uplink payload to preparing it for transmission to the backend.
  - T_{E3} is the instant when the VGW has processed the payload and is ready to send it to the backend.
- $D_{E3} = T_{E4} - T_{E3}$, representing the time taken for the uplink communication to travel from the VGW to the backend.
  - T_{E4} is the instant when the backend acknowledges the reception of the new uplink communication originating from the VGW.
- $D_{E4} = T_{E5} - T_{E4}$, representing the time taken for the uplink communication to forward from the backend to the NS.
  - T_{E5} identifies the instant when the NS is notified of the new uplink communication.
- $D_{E5} = T_{E6} - T_{E5}$, representing the additional time required for the uplink communication to travel from the NS to the AS.
  - T_{E6} identifies the instant when the AS is notified of the new uplink communication.
- $D_{EEE} = T_{E6} - T_{E1}$ is the End-to-End delay;
- $D_{END} = T_{E3} - T_{E1}$ is the Node delay;

Performance Overview

Tables 4.7–4.10 provide a summary of statistics for the time-related metrics, organized by different CPU (virtual) cores, with 8 GiB of memory allocated in all cases. Each table presents an overview of the performance metrics for specific intervals measured in milliseconds, allowing for an assessment of efficiency at each stage of the uplink message path across the virtualized environments and the overall path.

Additionally, Figures 4.12–4.15 illustrate the End-to-End and the Node uplink delays for different vCPU cores for all the configurations. All results presented in these figures are filtered to the 95th percentile in order to better highlight variations imputable to an increasing amount of computational resources, excluding upper outliers.

Obviously, metrics depending on the cloud-hosted backend are not affected by the changing number of vCPUs. On the other hand, it is also interesting to highlight that the time spent for processing the (de)modulation is not affected as well.

Table 4.7: Summary of time-related statistics for the Standard configuration.

CPU#	Statistic (ms)	D_{S1}	D_{S2}	D_{S3}	D_{S4}	End-to-End
CPU1	Mean	1.103	138.496	12.461	219.366	371.425
	Std	0.400	10.478	9.271	22.767	23.785
	Median	1.071	137.309	19.168	208.269	363.636
	Min	0.304	120.324	0.556	202.886	345.685
	Max	6.838	159.625	30.921	359.219	516.022
	5th Pctl	0.484	123.095	0.642	203.646	347.992
	95th Pctl	1.497	156.833	20.509	260.058	416.620
CPU3	Mean	0.376	138.174	12.210	216.399	367.159
	Std	0.092	10.574	8.907	18.713	19.706
	Median	0.362	137.456	18.889	207.070	361.409
	Min	0.140	120.250	0.560	203.083	343.897
	Max	1.295	161.385	26.186	298.126	451.522
	5th Pctl	0.316	122.813	0.633	203.580	347.655
	95th Pctl	0.467	156.853	19.678	260.683	410.096
CPU5	Mean	0.357	132.085	18.177	206.253	356.872
	Std	0.050	6.810	3.325	4.435	7.414
	Median	0.352	132.091	18.861	205.137	357.266
	Min	0.135	120.232	0.758	202.522	343.874
	Max	0.710	157.389	29.595	245.883	396.458
	5th Pctl	0.305	122.248	17.918	203.417	345.756
	95th Pctl	0.429	141.138	19.618	212.657	366.738
CPU7	Mean	0.365	136.863	12.528	206.501	356.256
	Std	0.049	11.034	9.340	4.184	7.567
	Median	0.366	135.949	19.508	205.521	356.014
	Min	0.186	119.126	0.636	202.592	343.174
	Max	0.744	159.278	41.206	231.535	406.053
	5th Pctl	0.318	120.957	0.693	203.455	345.833
	95th Pctl	0.446	155.653	20.201	215.862	366.599

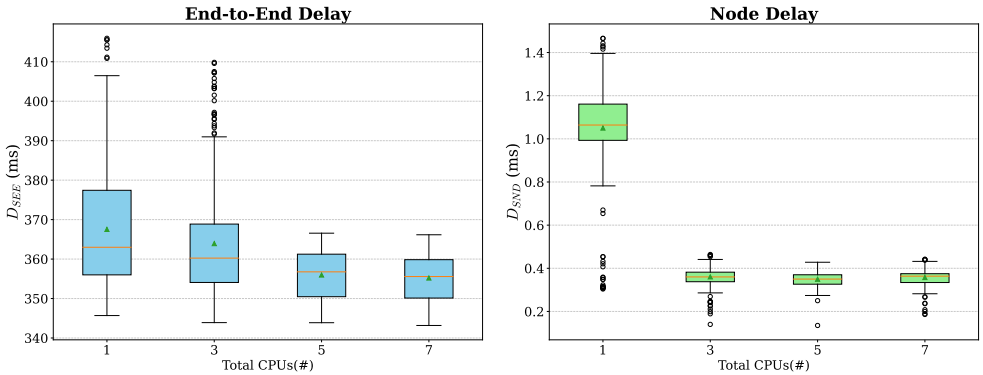


Figure 4.12: End-to-End and Node delays for different vCPU cores in the Standard configuration.

Table 4.8: Summary of time-related statistics for the Fully Virtualized (M-LoRa) configuration.

CPU#	Statistic (ms)	D_{FM1}	D_{FM2}	D_{FM3}	D_{FM4}	D_{FM5}	D_{FM6}	End-to-End
CPU1	Mean	3.077	63.747	28.435	1254.330	19.093	221.506	1590.188
	Std	0.757	22.839	7.938	66.981	1.008	42.424	84.692
	Median	2.977	57.090	29.614	1271.438	18.945	209.555	1604.391
	Min	0.157	49.896	16.147	1136.795	18.080	203.078	1448.641
	Max	6.942	338.310	109.459	1367.985	35.139	889.988	2254.066
	5th Pctl	2.691	54.845	17.019	1153.186	18.782	203.842	1468.904
	95th Pctl	3.809	102.914	36.966	1336.185	19.833	272.145	1694.090
CPU3	Mean	0.499	65.664	18.728	1258.625	19.037	212.172	1574.725
	Std	0.418	16.297	3.671	62.712	0.413	13.728	67.151
	Median	0.270	64.332	18.102	1275.234	18.930	206.596	1591.873
	Min	0.117	53.600	14.368	1142.156	18.729	203.102	1440.253
	Max	2.103	300.902	67.459	1347.932	23.384	290.774	1903.769
	5th Pctl	0.213	55.861	15.310	1156.576	18.797	203.727	1466.188
	95th Pctl	1.411	76.755	23.123	1332.375	19.744	244.304	1656.257
CPU5	Mean	0.273	65.453	17.390	1245.473	18.198	206.604	1553.391
	Std	0.126	13.885	2.904	73.764	0.453	3.778	75.676
	Median	0.219	65.186	17.367	1288.709	18.022	205.442	1592.710
	Min	0.089	55.081	13.886	1134.827	17.733	203.003	1434.621
	Max	0.894	248.150	62.538	1349.601	20.639	231.533	1842.127
	5th Pctl	0.199	56.918	14.865	1146.110	17.866	203.730	1450.479
	95th Pctl	0.602	77.618	19.945	1333.669	19.169	212.677	1642.595
CPU7	Mean	0.244	64.854	17.202	1238.718	18.332	207.290	1546.640
	Std	0.088	13.308	2.502	74.188	0.724	4.642	74.625
	Median	0.218	65.012	17.176	1271.380	18.352	206.605	1577.603
	Min	0.105	52.474	13.570	1138.849	17.443	203.579	1441.945
	Max	0.949	258.073	57.379	1355.924	21.150	258.361	1703.294
	5th Pctl	0.203	55.797	14.786	1147.882	17.519	203.972	1451.935
	95th Pctl	0.317	77.103	19.044	1331.395	19.560	212.428	1641.435

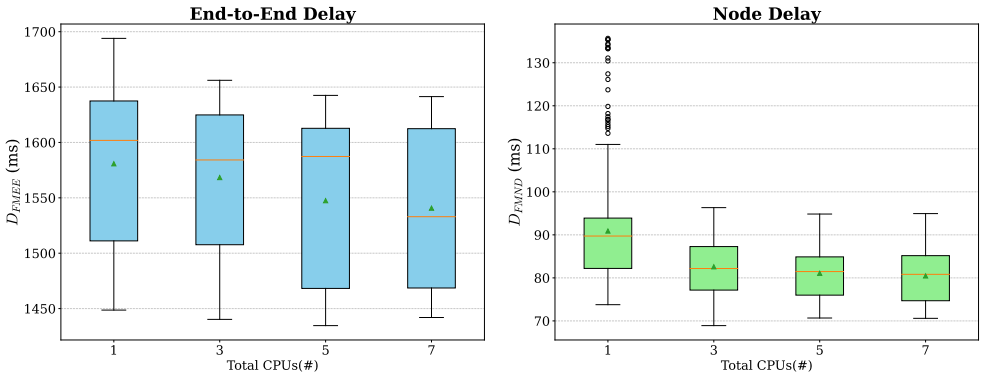


Figure 4.13: End-to-End and Node delays for different vCPU cores in the Fully Virtualized (M-LoRa) configuration.

4.2.9 Discussion

Any form of virtualization adds an additional software layer that translates into overhead in terms of computational and storage requirements. For this reason, the

Table 4.9: Summary of time-related statistics for the Fully Virtualized (C-LoRa) configuration.

CPU#	Statistic (ms)	D_{FC1}	D_{FC2}	D_{FC3}	D_{FC4}	D_{FC5}	D_{FC6}	End-to-End
CPU1	Mean	2.975	24.172	13.162	1261.525	19.994	215.693	1537.520
	Std	0.547	1.702	1.448	70.653	1.259	17.722	75.449
	Median	2.963	23.926	13.000	1289.530	19.791	207.085	1565.277
	Min	0.133	18.140	8.000	1132.531	19.565	203.063	1396.831
	Max	5.558	33.081	19.000	1362.278	35.340	296.815	1691.212
	5th Pctl	2.437	22.210	11.000	1152.206	19.648	203.912	1419.026
	95th Pctl	3.593	26.648	16.000	1348.793	20.702	257.668	1637.980
CPU3	Mean	0.400	26.451	7.673	1267.809	19.689	212.490	1534.513
	Std	0.368	2.963	2.558	68.490	1.187	14.757	71.085
	Median	0.251	25.908	8.000	1297.608	19.607	206.513	1561.665
	Min	0.083	15.525	3.000	1136.675	18.626	203.088	1395.617
	Max	2.428	58.778	18.000	1364.697	33.113	306.447	1668.700
	5th Pctl	0.211	23.987	4.000	1155.515	18.752	203.759	1417.508
	95th Pctl	1.228	31.007	11.150	1346.367	20.388	241.613	1624.121
CPU5	Mean	0.290	22.851	5.905	1259.138	19.360	206.594	1514.137
	Std	0.195	1.418	1.469	72.263	1.767	4.180	72.185
	Median	0.219	22.568	6.000	1299.792	19.178	205.624	1553.839
	Min	0.190	18.939	2.000	1138.660	17.921	203.061	1393.222
	Max	1.601	31.693	13.000	1352.913	49.466	244.842	1609.577
	5th Pctl	0.200	21.139	3.000	1152.890	18.062	203.780	1408.988
	95th Pctl	0.696	25.231	8.000	1339.763	20.401	212.012	1595.753
CPU7	Mean	0.236	22.656	6.043	1247.001	19.356	206.102	1501.395
	Std	0.059	1.193	1.235	72.868	0.838	3.073	72.973
	Median	0.221	22.410	6.000	1288.602	19.171	205.388	1541.454
	Min	0.142	20.725	2.000	1130.179	18.938	202.792	1387.329
	Max	0.744	30.936	10.000	1346.013	30.304	237.397	1598.844
	5th Pctl	0.205	21.167	4.000	1149.004	18.999	203.656	1402.240
	95th Pctl	0.288	24.467	8.000	1334.168	20.216	209.989	1588.620

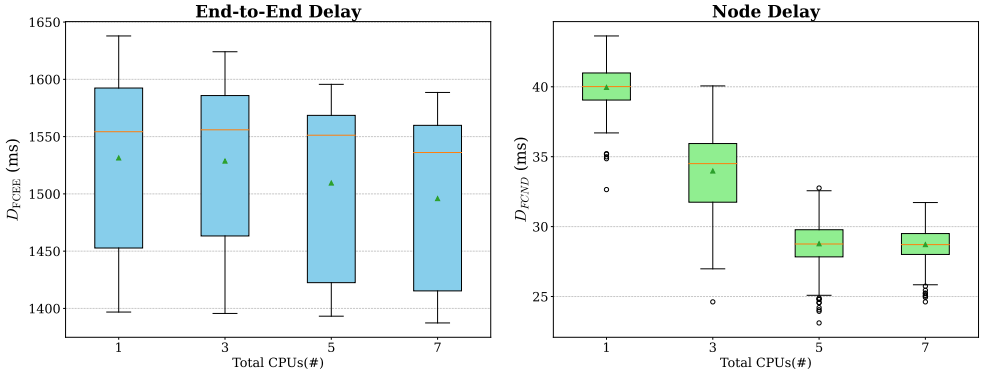


Figure 4.14: End-to-End and Node delays for different vCPU cores in the Fully Virtualized (C-LoRa) configuration.

proposed approach is not intended to substitute the use of full custom implementation of a LoRaWAN end device, which leverages the use of components explicitly designed to minimize the software and hardware footprints. On the other hand,

Table 4.10: Summary of time-related statistics for the Emulation configuration.

CPU#	Statistic (ms)	D_{E1}	D_{E2}	D_{E3}	D_{E4}	D_{E5}	End-to-End
CPU1	Mean	2.988	4.310	17.661	1.041	206.644	232.644
	Std	0.973	0.933	0.706	0.862	2.615	3.100
	Median	2.911	4.182	17.950	0.834	206.471	232.328
	Min	0.298	1.981	14.607	0.584	203.115	226.830
	Max	7.941	9.060	20.237	11.394	225.908	252.066
	5th Pctl	1.346	3.391	16.714	0.666	203.524	228.716
	95th Pctl	5.301	6.313	18.305	2.114	211.176	238.089
CPU3	Mean	0.516	6.950	18.090	1.382	207.329	234.266
	Std	0.351	3.087	1.705	4.954	4.189	7.333
	Median	0.347	6.900	17.687	0.821	206.565	233.118
	Min	0.156	1.137	15.566	0.591	202.669	223.405
	Max	2.243	13.362	25.468	85.180	239.199	313.516
	5th Pctl	0.296	2.033	16.048	0.652	203.606	227.222
	95th Pctl	1.334	11.380	19.917	2.047	216.235	244.661
CPU5	Mean	0.378	6.871	18.553	0.985	206.696	233.484
	Std	0.136	3.051	0.187	0.942	3.073	4.410
	Median	0.336	7.039	18.567	0.831	206.303	233.127
	Min	0.170	1.122	17.536	0.574	202.859	224.395
	Max	1.486	11.861	19.649	16.581	226.068	256.010
	5th Pctl	0.307	1.709	18.138	0.652	203.285	227.087
	95th Pctl	0.645	11.170	18.761	1.674	211.360	240.102
CPU7	Mean	0.370	6.990	19.715	0.989	206.958	235.021
	Std	0.105	3.011	0.351	0.665	4.592	5.544
	Median	0.333	7.185	19.679	0.834	206.496	234.724
	Min	0.199	1.052	19.016	0.599	202.755	225.999
	Max	1.201	11.653	23.148	9.318	265.534	296.767
	5th Pctl	0.310	1.908	19.440	0.655	203.519	228.480
	95th Pctl	0.601	11.018	19.920	1.892	211.362	240.759

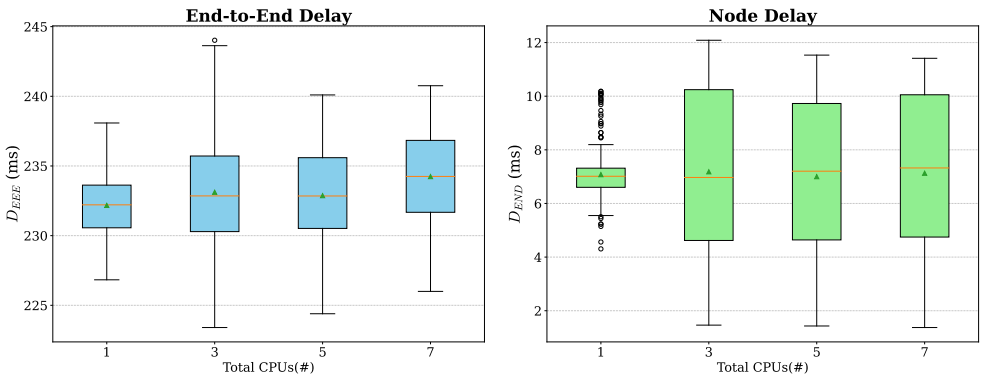


Figure 4.15: End-to-End and Node delays for different vCPU cores in the Emulation configuration.

these solutions are rigid and do not allow us to fully exploit the possibilities offered by the "white spaces" of LoRaWAN specifications. Radio virtualization enhances flexibility by enabling multiple (maybe heterogeneous) virtual radios to coexist on the same hardware. A well-accepted method for virtualizing radio devices involves the adoption of virtual machines and containers to multiplex software radio implementations over generic multipurpose radio hardware, as reported in [190].

For instance, regular LoRa radio transceivers, as the ones in the Semtech SX126X family, only permit the use of a single channel and SF at a time, preventing the implementation of interference/noise countermeasures based on the retransmission of the same message on multiple channels and/or multiple SFs. Indeed, efforts have been made to exploit SDR to support multi-user detection in LoRa radios based on successive interference cancellation and for integrating LoRa with ARQ error-control protocols and rate-less packet-level coding as in [178]. Software-defined transceivers also pave the way for an improved version of the LoRa scheme, e.g., as suggested in [191]. Another possible target application scenario is the addition of communication interfaces in systems where computational and storage resources are already available (as edge devices); in this case, the use of a full virtual approach would allow us to easily switch from one protocol to another without the need of changing the underlying hardware [192].

Another point that must be stressed is that recent research has focused on porting the containerization paradigm into constrained devices as well, allowing the integration of low-power IoT devices in DevOps workflows [193]. In this case, the efficient management of containers is also a main concern. For instance, the automotive industry is moving towards a software-defined model, which requires cars to host and execute a large number of applications with limited resources (in terms of limited memory, CPU, and energy); the solution could be the containerization of the applications, and energy and cost-efficiency could be reached exploiting the smart selection of containers to turn off based on container priorities and available resources, as suggested in [194].

A final comment is about the power consumption. The additional overhead introduced by virtualization not only requires more powerful hardware but also results in higher power consumption, if compared against a customized embedded solution. Additionally, SDRs are generally more power-hungry than regular hardwired transceivers, since the flexibility they offer also depends on higher bandwidth and sample rates, which adversely affect the supply current. For all these reasons, the proposed approach is suitable for applications where power supply is not an issue.

4.2.10 Conclusions

Our findings underline the substantial potential of container technologies to change the way nodes of wireless networks are implemented, even for relatively simple protocol stacks such as LoRaWAN. The transition to a fully virtualized design—where not only all layers of the LoRaWAN stack are disaggregated into containers, but also the radio is implemented by an SDR—proves particularly effective in enabling dy-

namic reconfiguration and efficient resource management, making it a promising solution for large-scale and flexible IoT deployments. This work not only confirms the feasibility of the proposed approach but also objectively quantifies the importance of an effective implementation, comparing the MATLAB- and C-based versions of the LoRa baseband. As expected, the C-based implementation provides better resource utilization, stability, and scalability, features of paramount importance in real-world applications.

While the results from this study are promising, the proposed architecture would benefit from further validation in field-based, dynamic environments to assess its performance when embedded platforms are considered. Additionally, integrating the containerized approach with other wireless communication technologies could provide insights into the broader applicability of this method in diverse IoT applications. This work paves the way for future innovations in wireless network virtualization, demonstrating the critical role of containerization in transforming IoT infrastructure management. Additional experiments have been already planned for a large-scale deployment to evaluate the impact of wireless channel impairments, which we currently intentionally limit by operating in a lab-grade, controlled environment.

Chapter 5

Conclusions

5.1 Contributions and key findings

The objective of this thesis was to investigate and propose enhancements to wireless communication technologies for IoT and vehicular communication. Among the candidate technologies for WWAN, Cellular-V2X and LoRaWAN were chosen for detailed exploration. These two technologies were analyzed with respect to their potential to address critical requirements in massive WWAN communication, particularly focusing on reliability, flexibility, and efficiency-three key performance metrics identified as crucial for the successful deployment of these systems.

The thesis pursued two primary goals: 1) to improve the reliability and efficiency of C-V2X communication for vehicle platooning, and 2) to enable flexible and resource-efficient IoT deployments through LoRaWAN virtualization. To achieve these goals, three research questions were formulated, each addressing specific aspects of C-V2X and LoRaWAN technologies. Answering these research questions allowed for a comprehensive characterization of the performance of these two technologies in the context of massive WWAN applications. It provided insights into their respective design parameters and the impact of these parameters on the key performance metrics. In particular, the enhanced SB-SPS algorithm for platooning (Paper I) and the virtualization of LoRaWAN nodes (Papers II and III) were proposed as solutions to improve reliability, resource efficiency, and scalability in both domains.

The results from this thesis can serve as valuable references for both designers and researchers, especially as the deployment of these technologies becomes more widespread. The findings reinforce the idea that both C-V2X and LoRaWAN have the potential to play a crucial role in the development of future WWAN systems. Following is a summary of the answers provided to the three research questions formulated in this work: **Research Question 1** How can the reliability of C-V2X communications be improved for vehicle platooning in OoC scenarios? The proposed dual-strategy mechanism, which integrates beacon-based time synchronization with partition-based scheduling, significantly improves the reliability of communication within platoons, even in OoC environments. By reducing resource collisions and minimizing update delays, this approach supports safer and more efficient platoon operations. **Research Question 2** Is it feasible to implement a fully software-defined LoRaWAN end node using lightweight container-based virtualization? Yes, the research demonstrates the feasibility of implementing a fully virtualized LoRaWAN node using container technologies. The modular architecture, where each layer of the LoRaWAN stack is deployed as a microservice in containers, enhances the flexibility and scalability of IoT deployments. However, throughput limitations between the SDR and the host system were identified as a key challenge. **Research Question 3** Can container-based virtualization be effectively used to virtualize LoRaWAN end nodes while maintaining performance and scalability in IoT deployments? The results show that container-based virtualization can effectively support LoRaWAN end nodes in terms of flexibility and resource efficiency. However, performance trade-offs related to resource usage and data throughput between virtualized components were observed. This calls for further optimization of the data handling be-

tween containers and physical interfaces to meet the latency and reliability requirements of IoT applications.

The research demonstrates that both technologies, though operating in different domains, can significantly benefit from advanced resource management and virtualization techniques. For C-V2X, future work could explore the integration of machine learning algorithms for dynamic scheduling and the transition of proposed methods to NR-V2X standards for improved scalability and performance. For LoRaWAN, addressing the data throughput limitations between containers and hardware interfaces will be critical to fully unlocking the benefits of containerized node architectures, particularly in embedded IoT platforms. Furthermore, exploring the synergy between containerization and edge computing could offer promising solutions for real-time processing and resource optimization, enabling large-scale IoT deployments.

While the virtualization of LoRaWAN nodes through containers shows promise, the performance of containerized solutions can be constrained by the resource limitations inherent in embedded platforms. Specifically, issues related to data throughput between containers and the hardware, particularly when interfacing with Software Defined Radios (SDRs), were observed. These limitations can significantly affect the overall performance, especially in scenarios requiring high throughput and low latency.

Future research should focus on:

- *Overcoming Throughput Bottlenecks:* Investigating more efficient communication protocols and interfaces between containers and physical hardware, especially for SDR-based systems, could mitigate the performance degradation observed in this study. Optimizing the data transfer mechanisms between the virtualized LoRaWAN stack and the underlying hardware is essential for improving the system's overall efficiency.
- *Optimization for Embedded Platforms:* As embedded systems are often resource-constrained, future research could explore lightweight containerization approaches tailored to these environments. This could involve optimizing memory usage, CPU allocation, and network management to ensure that LoRaWAN containerized nodes perform reliably without overburdening the platform.
- *Energy Efficiency:* In IoT applications, energy efficiency is a critical concern. Containerized LoRaWAN nodes need to be optimized for minimal energy consumption while maintaining functional performance. Research could investigate energy-efficient containerization techniques and resource allocation strategies that can reduce the operational costs of IoT networks.
- *Hybrid Approaches:* Combining containerization with edge computing could offer additional performance benefits. Edge nodes could offload more intensive computational tasks, leaving the LoRaWAN nodes to handle lighter, time-sensitive processes. This hybrid architecture could help address the limitations posed by embedded systems while maintaining the scalability and flexibility that containerization offers.

In conclusion, the findings from this thesis contribute to the understanding of LoRaWAN node virtualization using container technologies and the potential impact on IoT scalability and flexibility. However, further optimization is needed to fully exploit the benefits of containerization in embedded platforms. Addressing the current limitations will pave the way for more efficient and robust IoT systems, facilitating the deployment of large-scale, flexible, and resource-efficient networks in the future.

Together, the contributions of this thesis provide a solid foundation for the future development of robust, scalable, and resource-efficient communication systems in both vehicular networks and IoT infrastructures, paving the way for the next generation of wireless technologies.

Bibliography

[1] Hossein Khalilnasl, Salvatore Dello Iacono, Paolo Ferrari, Alessandra Flammini, Brian McCarthy, and Emiliano Sisinni. Enhancing c-v2x vehicle platooning: A dual-strategy of time synchronisation and partition-based scheduling. In *2024 IEEE International Symposium on Measurements & Networking (M&N)*, pages 1–6, 2024.

[2] Hossein Khalilnasl, Alessandro Depari, Paolo Ferrari, Alessandra Flammini, Massimiliano Gaffurini, and Emiliano Sisinni. Implementing a software defined lorawan node exploiting container-based lightweight virtualization. In *2024 IEEE International Symposium on Measurements & Networking (M&N)*, pages 1–6, 2024.

[3] Hossein Khalilnasl, Paolo Ferrari, Alessandra Flammini, and Emiliano Sisinni. On the use of containers for lorawan node virtualization: Practice and performance evaluation. *Electronics*, 14(8), 2025.

[4] Kais Mekki, Eddy Bajic, Frederic Chaxel, and Fernand Meyer. A comparative study of lpwan technologies for large-scale iot deployment. *ICT Express*, 5(1):1–7, 2019.

[5] Mahbubul Islam, Hossain Md Mubashshir Jamil, Samiul Ahsan Pranto, Rupak Kumar Das, Al Amin, and Arshia Khan. Future industrial applications: Exploring lpwan-driven iot protocols. *Sensors*, 24(8):2509, 2024.

[6] Mukarram A. M. Almuhaya, Waheb A. Jabbar, Noorazliza Sulaiman, and Suliman Abdulmalek. A survey on lorawan technology: Recent trends, opportunities, simulation tools and future directions. *Electronics*, 11(1), 2022.

[7] Mehzabien Iqbal, Abu Yousha Md Abdullah, and Farzana Shabnam. An application based comparative study of lpwan technologies for iot environment. In *2020 IEEE Region 10 Symposium (TENSYMP)*, pages 1857–1860, 2020.

[8] Gagandeep Kaur, Vipin Balyan, and Sindhu Hak Gupta. Experimental analysis of low-duty cycle campus deployed iot network using lora technology. *Results in Engineering*, 23:102844, 2024.

[9] Usman Raza, Parag Kulkarni, and Mahesh Sooriyabandara. Low power wide area networks: An overview. *IEEE Communications Surveys & Tutorials*, 19(2):855–873, 2017.

[10] Philani Larrance Ngwenyama and Ronald C. W. Webber-Youngman and. Recent advances, challenges and future trends for the applications of low power wide area networks (lpwans) technologies in underground mines. *International Journal of Mining, Reclamation and Environment*, 0(0):1–53, 2025.

[11] Stephen Ugwuanyi, Greig Paul, and James Irvine. Survey of iot for developing countries: Performance analysis of lorawan and cellular nb-iot networks. *Electronics*, 10(18), 2021.

[12] Safiu Abiodun Gbadamosi, Gerhard P. Hancke, and Adnan M. Abu-Mahfouz. Building upon nb-iot networks: A roadmap towards 5g new radio networks. *IEEE Access*, 8:188641–188672, 2020.

[13] Alper Yegin, Thorsten Kramp, Pierre Dufour, Rohit Gupta, Ramez Soss, Olivier Hersent, Derek Hunt, and Nicolas Sornin. 3 - lorawan protocol: specifications, security, and capabilities. In Bharat S. Chaudhari and Marco Zennaro, editors, *LPWAN Technologies for IoT and M2M Applications*, pages 37–63. Academic Press, 2020.

[14] LoRa Alliance. Lorawan network coverage, 2025. Accessed on: 19 April 2025.

[15] D. A. G. Alina Pinkelnig. Annual pilot overview report 2021. Technical report, C-Roads EU, August 2022.

[16] TSI. En 302 637-2 intelligent transport systems (its); vehicular communications; basic set of applications; part 2: Specification of cooperative awareness basic service (v1.4.1). Technical report, TSI, January 2019.

[17] TSI. Etsi en 302 637-3; intelligent transport systems (its); vehicular communications; basic set of applications; part 3: Specifications of decentralized environmental notification basic service (v1.3.1). Technical report, TSI, April 2019.

[18] TS ETSI. 103 613 intelligent transport systems (its): Access layer specification for intelligent transport systems using lte vehicle to everything communication in the 5, 9 ghz frequency band, v 1.1. 1, 2018.

[19] DSRC Technical Committee et al. *On-Board System Requirements for V2V Safety Communications*. SAE International, 2020.

[20] 3GPP. Study on lte-based v2x services (v14.0.0, release 14). Technical Report 3GPP TR 36.885, 3GPP, July 2016.

[21] 3GPP. Release 16 description; summary of rel-16 work items (v1.0.0, release 16). Technical Report 3GPP TR 21.916, 3GPP, December 2020.

[22] 3GPP. Proximity-based services (prose); stage 2. Technical Report 3GPP TS 23.303, 3GPP, March 2017.

[23] Brian McCarthy. *Modelling and Enhanced Scheduling in the Cellular Vehicular Sidelink Standards*. Phd thesis, University College Cork, 2024. Accessed on: 19 April 2025.

[24] Manuel Gonzalez-MartÃn, Miguel Sepulcre, Rafael Molina-Masegosa, and Javier Gozalvez. Analytical models of the performance of c-v2x mode 4 vehicular communications. *IEEE Transactions on Vehicular Technology*, 68(2):1155–1166, 2019.

[25] Alessandro Bazzi, Giammarco Cecchini, Alberto Zanella, and Barbara M. Masini. Study of the impact of phy and mac parameters in 3gpp c-v2v mode 4. *IEEE Access*, 6:71685–71698, 2018.

[26] Rafael Molina-Masegosa and Javier Gozalvez. System level evaluation of lte-v2v mode 4 communications and its distributed scheduling. In *2017 IEEE 85th Vehicular Technology Conference (VTC Spring)*, pages 1–5, 2017.

[27] 3GPP. Tr 36.213 physical layer procedures (v14.2.0, release 14). Technical Report 3GPP TS 36.213, 3GPP, December 2017.

[28] Mario H. Castaneda Garcia, Alejandro Molina-Galan, Mate Boban, Javier Gozalvez, Baldomero Coll-Perales, Taylan sahin, and Apostolos Kousaridas. A tutorial on 5g nr v2x communications. *IEEE Communications Surveys & Tutorials*, 23(3):1972–2026, 2021.

[29] Vittorio Todisco, Stefania Bartoletti, Claudia Campolo, Antonella Molinaro, Antoine O. Berthet, and Alessandro Bazzi. Performance analysis of sidelink 5g-v2x mode 2 through an open-source simulator. *IEEE Access*, 9:145648–145661, 2021.

[30] General Motors. The 1939 new york world's fair (1939), 1939. Accessed on: 3-04-2025.

[31] Assad Al Alam, Ather Gattami, and Karl Henrik Johansson. An experimental study on the fuel reduction potential of heavy duty vehicle platooning. In *13th International IEEE Conference on Intelligent Transportation Systems*, pages 306–311, 2010.

[32] Kuo-Yun Liang, Jonas MÃ¥rtensson, and Karl H. Johansson. Heavy-duty vehicle platoon formation for fuel efficiency. *IEEE Transactions on Intelligent Transportation Systems*, 17(4):1051–1061, 2016.

[33] Sebastian van de Hoef, Karl Henrik Johansson, and Dimos V. Dimarogonas. Fuel-efficient en route formation of truck platoons. *IEEE Transactions on Intelligent Transportation Systems*, 19(1):102–112, 2018.

[34] Valerio Turri, Bart Besselink, and Karl H. Johansson. Cooperative look-ahead control for fuel-efficient and safe heavy-duty vehicle platooning. *IEEE Transactions on Control Systems Technology*, 25(1):12–28, 2017.

[35] Pedro Fernandes and Urbano Nunes. Multiplatooning leaders positioning and cooperative behavior algorithms of communicant automated vehicles for high traffic capacity. *IEEE Transactions on Intelligent Transportation Systems*, 16(3):1172–1187, 2015.

[36] Farhad Farokhi and Karl H. Johansson. A study of truck platooning incentives using a congestion game. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):581–595, 2015.

[37] Can Zhao, Li Li, Jingwei Li, Keqiang Li, and Zhiheng Li. The impact of truck platoons on the traffic dynamics around off-ramp regions. *IEEE Access*, 9:57010–57019, 2021.

[38] Assad Alam, Bart Besselink, Valerio Turri, Jonas MÃ¥Rtensson, and Karl H. Johansson. Heavy-duty vehicle platooning for sustainable freight transportation: A cooperative method to enhance safety and efficiency. *IEEE Control Systems Magazine*, 35(6):34–56, 2015.

[39] Steven E. Shladover. Path at 20-history and major milestones. *IEEE Transactions on Intelligent Transportation Systems*, 8(4):584–592, 2007.

[40] O. Gehring and H. Fritz. Practical results of a longitudinal control concept for truck platooning with vehicle to vehicle communication. In *Proceedings of Conference on Intelligent Transportation Systems*, pages 117–122, 1997.

[41] Cristofer Englund, Lei Chen, Jeroen Ploeg, Elham Semsar-Kazerooni, Alexey Voronov, Hoai Hoang Bengtsson, and Jonas Didoff. The grand cooperative driving challenge 2016: boosting the introduction of cooperative automated vehicles. *IEEE Wireless Communications*, 23(4):146–152, 2016.

[42] Jeroen Ploeg, Cristofer Englund, Henk Nijmeijer, Elham Semsar-Kazerooni, Steven E. Shladover, Alexey Voronov, and Nathan van de Wouw. Guest editorial introduction to the special issue on the 2016 grand cooperative driving challenge. *IEEE Transactions on Intelligent Transportation Systems*, 19(4):1208–1212, 2018.

[43] Jeroen Ploeg, Steven Shladover, Henk Nijmeijer, and Nathan van de Wouw. Introduction to the special issue on the 2011 grand cooperative driving challenge. *IEEE Transactions on Intelligent Transportation Systems*, 13(3):989–993, 2012.

[44] ENSEMBLE. The h2020 project ensemble, 2025. Last accessed on: 03-04-2025.

[45] Alexey Vinel, Lin Lan, and Nikita Lyamin. Vehicle-to-vehicle communication in c-acc/platooning scenarios. *IEEE Communications Magazine*, 53(8):192–197, 2015.

[46] Hao Liu, Xiao-Yun Lu, and Steven E. Shladover. Traffic signal control by leveraging cooperative adaptive cruise control (cacc) vehicle platooning capabilities. *Transportation Research Part C: Emerging Technologies*, 104:390–407, 2019.

[47] Chris Nowakowski, Steven E. Shladover, Xiao-Yun Lu, and Dale Thompson. Cooperative adaptive cruise control (cacc) for truck platooning: Operational concept alternatives. Technical report, University of California, Berkeley, 2015.

[48] Ziran Wang, Guoyuan Wu, Peng Hao, Kanok Boriboonsomsin, and Matthew Barth. Developing a platoon-wide eco-cooperative adaptive cruise control (cacc) system. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1256–1261, 2017.

[49] Danjue Chen, Soyoung Ahn, Madhav Chitturi, and David Noyce. Truck platooning on uphill grades under cooperative adaptive cruise control (cacc). *Transportation Research Part C: Emerging Technologies*, 94:50–66, 2018. ISTTT22.

[50] Yiming Zhang, Zhizhou Wu, Yu Zhang, Zhiying Shang, Ping Wang, Qingquan Zou, Xianhong Zhang, and Jia Hu. Human-lead-platooning cooperative adaptive cruise control. *IEEE Transactions on Intelligent Transportation Systems*, 23(10):18253–18272, 2022.

[51] Songtao Xie, Junyan Hu, Zhengtao Ding, and Farshad Arvin. Cooperative adaptive cruise control for connected autonomous vehicles using spring damping energy model. *IEEE Transactions on Vehicular Technology*, 72(3):2974–2987, 2023.

[52] Chaojie Wang, Siyuan Gong, Anye Zhou, Tao Li, and Srinivas Peeta. Cooperative adaptive cruise control for connected autonomous vehicles by factoring communication-related constraints. *Transportation Research Part C: Emerging Technologies*, 113:124–145, 2020. ISTTT 23 TR_C-23rd International Symposium on Transportation and Traffic Theory (ISTTT 23).

[53] Wei Gao, Celimuge Wu, Baozhu Li, and Siri Guleng. Communication resource allocation in platooning management based on c-v2x with spectrum sensing. In *2021 International Conference on Information and Communication Technologies for Disaster Management (ICT-DM)*, pages 15–21, 2021.

[54] Ruyan Wang, Jiaxing Wu, and Junjie Yan. Resource allocation for d2d-enabled communications in vehicle platooning. *IEEE Access*, 6:50526–50537, 2018.

[55] Mehdi Harounabadi, Dariush Mohammad Soleymani, Shubhangi Bhadauria, Martin Leyh, and Elke Roth-Mandutz. V2x in 3gpp standardization: Nr sidelink in release-16 and beyond. *IEEE Communications Standards Magazine*, 5(1):12–21, 2021.

[56] Rafael Molina-Masegosa and Javier Gozalvez. Lte-v for sidelink 5g v2x vehicular communications: A new 5g technology for short-range vehicle-to-everything communications. *IEEE Vehicular Technology Magazine*, 12(4):30–39, 2017.

[57] Amr Nabil, Komalbir Kaur, Carl Dietrich, and Vuk Marojevic. Performance analysis of sensing-based semi-persistent scheduling in c-v2x networks. In *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, pages 1–5, 2018.

[58] Rafael Molina-Masegosa, Javier Gozalvez, and Miguel Sepulcre. Configuration of the c-v2x mode 4 sidelink pc5 interface for vehicular communication. In *2018 14th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN)*, pages 43–48, 2018.

[59] Stefania Bartoletti, Barbara Mavi Masini, Vincent Martinez, Ioannis Sarris, and Alessandro Bazzi. Impact of the generation interval on the performance of sidelink c-v2x autonomous mode. *IEEE Access*, 9:35121–35135, 2021.

[60] Luis F. Abanto-Leon, Arie Koppelaar, and Sonia Heemstra de Groot. Enhanced c-v2x mode-4 subchannel selection. In *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, pages 1–5, 2018.

[61] Jiaqi Zhao, Xinxin He, Huan Wang, Xufei Zheng, Jie Lv, Tao Luo, and Xiaolin Hou. Cluster-based resource selection scheme for 5g v2x. In *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, pages 1–5, 2019.

[62] Rafael Molina-Masegosa, Miguel Sepulcre, and Javier Gozalvez. Geo-based scheduling for c-v2x networks. *IEEE Transactions on Vehicular Technology*, 68(9):8397–8407, 2019.

[63] Alessandro Bazzi, Claudia Campolo, Antonella Molinaro, Antoine O. Berthet, Barbara M. Masini, and Alberto Zanella. On wireless blind spots in the c-v2x sidelink. *IEEE Transactions on Vehicular Technology*, 69(8):9239–9243, 2020.

[64] Yongseok Jeon and Hyogon Kim. An explicit reservation-augmented resource allocation scheme for c-v2x sidelink mode 4. *IEEE Access*, 8:147241–147255, 2020.

[65] Xinxin He, Jie Lv, Jiaqi Zhao, Xiaolin Hou, and Tao Luo. Design and analysis of a short-term sensing-based resource selection scheme for c-v2x networks. *IEEE Internet of Things Journal*, 7(11):11209–11222, 2020.

[66] Saif Sabeeh, Pawel Sroka, and Krzysztof Wesolowski. Estimation and reservation for autonomous resource selection in c-v2x mode 4. In *2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pages 1–6, 2019.

[67] Saif Sabeeh and Krzysztof Wesolowski. C-v2x mode 4 resource allocation in high mobility vehicle communication. In *2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications*, pages 1–6, 2020.

[68] Jin Mo Yang, Hoyoung Yoon, Sunwook Hwang, and Saewoong Bahk. Press: Predictive assessment of resource usage for c-v2v mode 4. In *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, 2021.

[69] Brian McCarthy and Aisling O'Driscoll. Opencv2x mode 4: A simulation extension for cellular vehicular communication networks. In *2019 IEEE 24th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pages 1–6, 2019.

[70] Mario H. Castaneda Garcia, Alejandro Molina-Galan, Mate Boban, Javier Gozalvez, Baldomero Coll-Perales, Taylan sahin, and Apostolos Kousaridas. A tutorial on 5g nr v2x communications. *IEEE Communications Surveys & amp; Tutorials*, 23(3):1972–2026, 2021.

[71] Alessandro Bazzi, Antoine O. Berthet, Claudia Campolo, Barbara Mavi Masini, Antonella Molinaro, and Alberto Zanella. On the design of sidelink for cellular v2x: A literature review and outlook for future. *IEEE Access*, 9:97953–97980, 2021.

[72] Anis Boubakri and Sonia Mettali Gammar. Inter-platoons communication in autonomous vehicles: A survey. In *2022 International Wireless Communications and Mobile Computing (IWCMC)*, pages 1154–1159, 2022.

[73] Guanhua Chai, Weihua Wu, Qinghai Yang, Meng Qin, Yan Wu, and F. Richard Yu. Platoon partition and resource allocation for ultra-reliable v2x networks. *IEEE Transactions on Vehicular Technology*, 73(1):147–161, 2024.

[74] Annu and P. Rajalakshmi. Distance-based queue modeling in sidelink c-v2x communication for platooning: Towards 6g-v2x. In *2024 IEEE 100th Vehicular Technology Conference (VTC2024-Fall)*, pages 1–5, 2024.

[75] Ameur Lelio Chelli, Maria Luisa Merani, Luca Lusvarghi, Federico Pasquinucci, Riccardo DonÃ , Konstantinos Mattas, and Biagio Ciuffo. Nr-v2x for cooperative adaptive cruise control. In *2024 IEEE 100th Vehicular Technology Conference (VTC2024-Fall)*, pages 1–7, 2024.

[76] Giovanni Giambene, MD Saifur Rahman, and Alexey Vinel. Analysis of v2v sidelink communications for platoon applications. In *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, pages 1–6, 2020.

[77] Anis Boubakri and Sonia Mettali Gammar. Enhancing platoon performance: A novel approach to speed and direction control using v2x communication. *International Journal of Knowledge-Based and Intelligent Engineering Systems*, 28(3):517–537, 2024.

[78] Junhyeong Kim, Youngnam Han, and Ilgyu Kim. Efficient groupcast schemes for vehicle platooning in v2v network. *IEEE Access*, 7:171333–171345, 2019.

[79] Xin Gu, Jun Peng, Lin Cai, Weirong Liu, Xiaoyong Zhang, and Zhiwu Huang. Resource reservation coordination for vehicle platooning in c-v2x networks. *IEEE Transactions on Wireless Communications*, 23(6):5515–5528, 2024.

[80] Michele Segata, Piermaria Arvani, and Renato Lo Cigno. A critical assessment of c-v2x resource allocation scheme for platooning applications. In *2021 16th Annual Conference on Wireless On-demand Network Systems and Services Conference (WONS)*, pages 1–8, 2021.

[81] Bingying Wang, Jun Zheng, Nathalie Mitton, and Cheng Li. An enhanced c-v2x mode 4 resource selection scheme for cav platoons in a multilane highway scenario. In *2023 International Conference on Future Communications and Networks (FCN)*, pages 1–6, 2023.

[82] Andres Villamil, Arturo Gonzalez, and Gerhard Fettweis. Optimal packet transmission rates for platooning under random access c-v2x. In *2022 IEEE 96th Vehicular Technology Conference (VTC2022-Fall)*, pages 1–5, 2022.

[83] Xiaobo Wang, Qiong Wu, Pingyi Fan, Qiang Fan, Huiling Zhu, and Jiangzhou Wang. Vehicle selection for c-v2x mode 4-based federated edge learning systems. *IEEE Systems Journal*, 18(4):1927–1938, 2024.

[84] Gulabi Mandal, Anik Roy, and Basabdatta Palit. Performance analysis of resource allocation algorithms for vehicle platoons over 5g ev2x communication, 2024.

[85] Bingying Wang, Jun Zheng, and Nathalie Mitton. A packet collision avoidance resource selection scheme for reliable intra-platoon message delivery in a c-v2x network. *IEEE Transactions on Vehicular Technology*, 2025. DOI: `https://hal.archives-ouvertes.fr/hal-04904559`.

[86] Zhiyu Shao, Qiong Wu, Pingyi Fan, Kezhi Wang, Qiang Fan, Wen Chen, and Khaled B. Letaief. Semantic-aware resource management for c-v2x platooning via multi-agent reinforcement learning, 2024.

[87] Naila Bouchemal and Jae-Yun Jun. V2x architecture for autonomous platoon management in urban environment. In *2021 International Conference on Computer, Information and Telecommunication Systems (CITS)*, pages 1–6, 2021.

[88] Taesik Nam, Seungjae Lee, Nathan Jeong, Han-Shin Jo, and Jong-Gwan Yook. Proactive resource allocation in c-v2x for cacc-based platoon driving service. In *2024 IEEE 100th Vehicular Technology Conference (VTC2024-Fall)*, pages 1–5, 2024.

[89] Xingquan Wang, Yongfu Li, Longwang Huang, Xin Huang, Hang Zhao, Xingjun Han, Jie Xiang, and Huan Wang. Cooperative control for connected automated vehicle platoon with c-v2x pc5 interface. In *2023 IEEE International Conference on Unmanned Systems (ICUS)*, pages 635–640, 2023.

[90] Khabaz Sehla, Thi Mai Trang Nguyen, Guy Pujolle, and Pedro Braconnot Velloso. Resource allocation modes in c-v2x: From lte-v2x to 5g-v2x. *IEEE Internet of Things Journal*, 9(11):8291–8314, 2022.

[91] Anis BOUBAKRI and Sonia METTALI GAMMAR. Intra-platoon communication in autonomous vehicle: A survey. In *2020 9th IFIP International Conference on Performance Evaluation and Modeling in Wireless Networks (PEMWN)*, pages 1–6, 2020.

[92] Haizhou Bao and Yiming Huo. Platoon-based resource allocation in noma-integrated v2x networks. In *2023 IEEE 23rd International Conference on Communication Technology (ICCT)*, pages 821–826, 2023.

[93] Xin Gu, Jun Peng, Lin Cai, Xiaoyong Zhang, and Zhiwu Huang. Markov analysis of c-v2x resource reservation for vehicle platooning. In *2022 IEEE 95th Vehicular Technology Conference: (VTC2022-Spring)*, pages 1–5, 2022.

[94] Wei Gao, Yan Shi, and Shanzhi Chen. Proactive platooning based on c-v2x to relieve congestion at a signalized intersection. *China Communications*, 20(2):155–167, 2023.

[95] Ruirui Ning, Xiaokang Zhang, Weiyang Feng, Ning Zhang, and Siyu Lin. Performance analysis of vehicle platoon communication in c-v2x autonomous mode. In *2022 IEEE 23rd International Conference on High Performance Switching and Routing (HPSR)*, pages 41–46, 2022.

[96] Giammarco Cecchini, Alessandro Bazzi, Barbara M. Masini, and Alberto Zanella. Ltev2vsim: An lte-v2v simulator for the investigation of resource allocation for cooperative awareness. In *2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, pages 80–85, 2017.

[97] Alessandro Bazzi. Congestion control mechanisms in ieee 802.11p and sidelink c-v2x. In *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, pages 1125–1130, 2019.

[98] Marianna Gois de Campos, Lahis Gomes de Almeida, Luiz Eduardo Mendes Matheus, and Juliana Freitag Borin. On the simulation of lorawan networks: A focus on reproducible parameter configuration. *Computer Networks and Communications*, pages 148–171, 2024.

[99] Roger Pueyo Centelles, Felix Freitag, Roc Meseguer, and Leandro Navarro. Beyond the star of stars: An introduction to multihop and mesh for lora and lorawan. *IEEE Pervasive Computing*, 20(2):63–72, 2021.

[100] Wilbur Myrick, Nobuyasu Shiga, Julian St. James, and Ahmad Byagowi. Timing, communications, and ranging sdr (tcr-sdr) for iot wireless synchronization. In *2024 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*, pages 1–7, 2024.

[101] Lorenzo Vangelista and Marco Centenaro. Worldwide connectivity for the internet of things through lorawan. *Future Internet*, 11(3):57, 2019.

[102] The Things Network. The things network, 2025. Accessed: 2025-04-29.

[103] Stefanie Ziltener. Security improvements for object tracking using lorawan and the things network. Master's thesis, University of Zurich, Zurich, Switzerland, 2018.

[104] Behnaz Asadollahseraj. *Design of Low Power Gateways for LoRaWAN Applications In Remote Areas*. PhD thesis, Politecnico di Torino, 2018.

[105] Juha PetÃ¤jÃ¤jÃ¤rvi, Konstantin Mikhaylov, Marko Pettissalo, Janne Janhunen, and Jari Iinatti. Performance of a low-power wide-area network based on lora technology: Doppler robustness, scalability, and coverage. *International Journal of Distributed Sensor Networks*, 13(3):1550147717699412, 2017.

[106] LoRa Alliance. Lorawan® specification v1.0.3, 2025. Accessed on: 03-04-2025.

[107] LoRa Alliance. Lorawan® 1.0.4 specification package, 2025. Accessed on: 03-04-2025.

[108] LoRa Alliance. Lorawan® specification v1.1, 2025. Accessed on: 03-04-2025.

[109] LoRa Alliance. Lorawan specification version 1.0.3. Technical report, LoRa Alliance, 2018.

[110] LoRa Alliance. TS001-1.0.4 lorawan l2 1.0.4 specification. Technical report, LoRa Alliance, 2020.

[111] Joachim Tapparel. Complete reverse engineering of lora phy. Technical report, Telecommunications Circuits Laboratory, EPFL, 2020.

[112] Wikipedia contributors. Lorawan — wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/LoRaWAN`, 2025. Accessed: 2025-04-26.

[113] LoRa Alliance. Adr harmonize state transition errata on the lorawan l2 1.0.3 specification. Technical report, LoRa Alliance, 2018.

[114] R. R. Yadav, E. T. G. Sousa, and G. R. A. Callou. Performance comparison between virtual machines and docker containers. *IEEE Latin America Transactions*, 16(8):2282–2288, 2018.

[115] Qi Zhang, Ling Liu, Calton Pu, Qiwei Dou, Liren Wu, and Wei Zhou. A comparative study of containers and virtual machines in big data environment. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pages 178–185, 2018.

[116] Roberto Morabito, Jimmy KjÃ¤llman, and Miika Komu. Hypervisors vs. lightweight virtualization: A performance comparison. In *2015 IEEE International Conference on Cloud Engineering*, pages 386–393, 2015.

[117] Matt Helsley. Lxc: Linux container tools. *IBM devloperWorks Technical Library*, 11, 2009.

[118] Dirk Merkel et al. Docker: lightweight linux containers for consistent development and deployment. *Linux j*, 239(2):2, 2014.

[119] A. Anand and A. Nisha Jebaseeli. A comparative analysis of virtual machines and containers using queuing models. *The Scientific Temper*, 15(spl-1):1–7, 2024.

[120] Daniel Silva, JoÃ£o Rafael, and Alexandre Fonte. Toward optimal virtualization: An updated comparative analysis of docker and lxd container technologies. *Computers*, 13(4), 2024.

[121] Lubomir Urblik, Erik Kajati, Peter Papcun, and Iveta ZolotovÃ¡. Containerization in edge intelligence: A review. *Electronics*, 13(7), 2024.

[122] Kalvin Eng, Abram Hindle, and Eleni Stroulia. Patterns of multi-container composition for service orchestration with docker compose. *Empirical Software Engineering*, 29(3):65, 2024.

[123] David Bernstein. Containers and cloud: From lxc to docker to kubernetes. *IEEE Cloud Computing*, 1(3):81–84, 2014.

[124] Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Niels Bouten, Filip De Turck, and Raouf Boutaba. Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys & Tutorials*, 18(1):236–262, 2016.

[125] Emiliano Casalicchio. *Container Orchestration: A Survey*, pages 221–235. Springer International Publishing, Cham, 2019.

[126] Nteziriza Nkerabahizi Josbert, Min Wei, Ping Wang, and Ahsan Rafiq. A look into smart factory for industrial iot driven by sdn technology: A comprehensive survey of taxonomy, architectures, issues and future research orientations. *Journal of King Saud University - Computer and Information Sciences*, 36(5):102069, 2024.

[127] Umair Shafiq Khan and Tahira Mahboob. Network softwarization and virtualization: Management of qos in wireless and mobile networks. In Yaseein Soubhi Hussein and Abdulmajeed Al-Jumaily, editors, *Quality of Service (QoS)*, chapter 2. IntechOpen, Rijeka, 2024.

[128] Amr Adel, Noor HS Alani, and Tony Jan. Factories of the future in industry 5.0-softwarization, servitization, and industrialization. *Internet of Things*, 28:101431, 2024.

[129] Anurag Satpathy, Manmath Narayan Sahoo, Chittaranjan Swain, Paolo Bellavista, Mohsen Guizani, Khan Muhammad, and Sambit Bakshi. Virtual network embedding: Literature assessment, recent advancements, opportunities, and challenges. *IEEE Communications Surveys & Tutorials*, pages 1–1, 2025.

[130] Tianzhu Zhang, Han Qiu, Leonardo Linguaglossa, Walter Cerroni, and Paolo Giaccone. Nfv platforms: Taxonomy, design choices and future challenges. *IEEE Transactions on Network and Service Management*, 18(1):30–48, 2021.

[131] Hassan Hawilo, Abdallah Shami, Maysam Mirahmadi, and Rasool Asal. Nfv: state of the art, challenges, and implementation in next generation mobile networks (vepc). *IEEE Network*, 28(6):18–26, 2014.

[132] Jing Su, Suku Nair, and Leo Popokh. Optimal resource allocation in sdn/nfv-enabled networks via deep reinforcement learning. In *2022 IEEE Ninth International Conference on Communications and Networking (ComNet)*, pages 1–7, 2022.

[133] Iqbal Alam, Kashif Sharif, Fan Li, Zohaib Latif, Md Monjurul Karim, Sujit Biswas, Boubakr Nour, and Yu Wang. A survey of network virtualization techniques for internet of things using sdn and nfv. *ACM Computing Surveys (CSUR)*, 53(2):1–40, 2020.

[134] Wei Liu, Joao F Santos, Jonathan van de Belt, Xianjun Jiao, Ingrid Moerman, Johann Marquez-Barja, Luiz DaSilva, and Sofie Pollin. Enabling virtual radio functions on software defined radio for future wireless networks. *Wireless Personal Communications*, 113:1579–1595, 2020.

[135] Maicon Kist, Juergen Rochol, Luiz A. DaSilva, and Cristiano Bonato Both. Hydra: A hypervisor for software defined radios to enable radio virtualization in mobile networks. In *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 960–961, 2017.

[136] Ederson Ribas Machado, Max Feldman, and Ivan MÃ¼ller. A container-based architecture to provide services from sdr devices. In *2023 IEEE 21st International Conference on Industrial Informatics (INDIN)*, pages 1–6, 2023.

[137] Felipe A. P. de Figueiredo, Xianjun Jiao, Wei Liu, and Ingrid Moerman. Radio hardware virtualization for software-defined wireless networks. *Wireless Personal Communications*, 100:113–126, 2018.

[138] Analog Devices. Adalm-pluto software-defined radio active learning module. `https://www.analog.com/en/resources/evaluation-hardware-and-software/evaluation-boards-kits/adalm-pluto.html`, 2025. Accessed: 2025-02-04.

[139] Samer Baher Safa Hanbali. Low-cost software-defined radio for electrical engineering education. *IEEE Potentials*, 42(5):13–19, 2023.

[140] Eryk Schiller, Silas Weber, and Burkhard Stiller. Design and evaluation of an sdr-based lora cloud radio access network. In *2020 16th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 1–7, 2020.

[141] Iman Ebrahimi Mehr, Alex Minetto, Fabio Dovis, Emanuele Pica, Claudio Cesaroni, and Vincenzo Romano. An open architecture for signal monitoring and recording based on sdr and docker containers: A gnss use case. In *IEEE EUROCON 2023-20th International Conference on Smart Technologies*, pages 66–71. IEEE, 2023.

[142] Johannes K Becker and David Starobinski. Snout: A middleware platform for software-defined radios. *IEEE Transactions on Network and Service Management*, 20(1):644–657, 2022.

[143] Stefan Gvozdenovic, Johannes K Becker, and David Starobinski. Sdr-based phy characterization of zigbee devices. In *2020 IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 129–132, 2020.

[144] Intidhar Bedhief, Meriem Kassar, and Taoufik Aguili. Empowering sdn-docker based architecture for internet of things heterogeneity. *Journal of Network and Systems Management*, 31(1):14, 2023.

[145] Hossein Aqasizade, Ehsan Ataie, and Mostafa Bastam. Experimental assessment of containers running on top of virtual machines. *IET Networks*, 14(1), 2024.

[146] Shatha A. Baker, Hesham Hashim Mohammed, and Omar I. Alsaif. Docker container security analysis based on virtualization technologies. *International Journal for Computers & Their Applications*, 31(1):69–78, 2024.

[147] Prathamesh Muzumdar, Amol Bhosale, Ganga Prasad Basyal, and George Kurian. Navigating the docker ecosystem: A comprehensive taxonomy and survey. *Asian Journal of Research in Computer Science*, 17(1):42–61, 2024.

[148] Antonio Cilfone, Luca Davoli, and Gianluigi Ferrari. Lora meets ip: A container-based architecture to virtualize lorawan end nodes. *IEEE Transactions on Mobile Computing*, 23(10):9191–9207, 2024.

[149] Hossein Khalilnasl, Alessandro Depari, Paolo Ferrari, Alessandra Flammini, Massimiliano Gaffurini, and Emiliano Sisinni. Implementing a software defined lorawan node exploiting container-based lightweight virtualization. In *2024 IEEE International Symposium on Measurements & Networking (M&N)*, pages 1–6. IEEE, 2024.

[150] Massimiliano Gaffurini, Alessandra Flammini, Paolo Ferrari, Dhiego Fernandes Carvalho, Eduardo Paciencia Godoy, and Emiliano Sisinni. End-to-end emulation of lorawan architecture and infrastructure in complex smart city scenarios exploiting containers. *Sensors*, 24(7):2024, 2024.

[151] Emiliano Sisinni, Alessandra Flammini, Massimiliano Gaffurini, Marco Pasetti, Stefano Rinaldi, and Paolo Ferrari. Lorawan end device disaggregation and decomposition by means of lightweight virtualization. *Internet of Things*, 25:101033, 2024.

[152] Brian McCarthy, Andres Burbano-Abril, Victor Rangel Licea, and Aisling O'Driscoll. Opencv2x: Modelling of the v2x cellular sidelink and performance evaluation for aperiodic traffic, 2021.

[153] 3GPP. Lte; evolved universal terrestrial radio access (e-utra); radio resource control (rrc); protocol specification (3gpp ts 36.331 version 14.16.0 release 14). Technical Specification (TS) 36.331, 3rd Generation Partnership Project (3GPP), 01 2021. V. 14.16.0.

[154] TCITS ETSI. Intelligent transport systems (its); vehicular communications; basic set of applications; part 2: Specification of cooperative awareness basic service. *Draft ETSI TS*, 20(2011):448–51, 2011.

[155] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018.

[156] Michele Segata, Renato Lo Cigno, Tobias Hardes, Julian Heinovski, Max Schettler, Bastian Bloessl, Christoph Sommer, and Falko Dressler. Multi-Technology Cooperative Driving: An Analysis Based on PLEXE. *IEEE Transactions on Mobile Computing (TMC)*, 2 2022. to appear.

[157] Christoph Sommer, Reinhard German, and Falko Dressler. Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis. *IEEE Trans. on Mobile Computing (TMC)*, 10(1):3–15, 1 2011.

[158] Jeroen Ploeg, B.T.M. Scheepers, E. van Nunen, N. van de Wouw, and H. Nijmeijer. Design and Experimental Evaluation of Cooperative Adaptive Cruise Control. In *14th IEEE Int. Conf. on Intelligent Transportation Systems (ITSC 2011)*, pages 260–265, 10 2011.

[159] Antonio Virdis, Giovanni Stea, and Giovanni Nardini. SimuLTE - A Modular System-level Simulator for LTE/LTE-A Networks based on OMNeT++. In *4th Int. Conf. on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2014)*, Vienna, AT, 8 2014.

[160] Andras Varga. *OMNeT++*, pages 35–59. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[161] Christoph Sommer, Reinhard German, and Falko Dressler. Bidirectionally coupled network and road traffic simulation for improved ivc analysis. *IEEE Transactions on Mobile Computing*, 10(1):3–15, 2011.

[162] Levente Mészáros, Andras Varga, and Michael Kirsche. *INET Framework*, pages 55–106. Springer International Publishing, Cham, 2019.

[163] Antonio Virdis, Giovanni Stea, and Giovanni Nardini. Simulating lte/lte-advanced networks with simulte. In Mohammad S. Obaidat, Tuncer Ören, Janusz Kacprzyk, and Joaquim Filipe, editors, *Simulation and Modeling Methodologies, Technologies and Applications*, pages 83–105, Cham, 2015. Springer International Publishing.

[164] 3rd Generation Partnership Project (3GPP). TS 36.323: Evolved universal terrestrial radio access (e-utra); packet data convergence protocol (pdcp) specification. Technical Specification 36.323, 3GPP, Valbonne, France, 2023. Version 16.7.0, Release 16.

[165] 3rd Generation Partnership Project (3GPP). TS 36.322: Evolved universal terrestrial radio access (e-utra); radio link control (rlc) protocol specification. Technical Specification 36.322, 3GPP, Valbonne, France, 2020. Version 16.0.0, Release 16.

[166] 3GPP. 3gpp technical specification group services and system aspects; system architecture for the 5g system (5gs) (release 12). Technical Report TS 23.501, 3rd Generation Partnership Project (3GPP), 2015. Accessed: 2025-05-02.

[167] Brian McCarthy, Andres Burbano-Abril, Víctor Rangel-Licea, and Aisling O'Driscoll. Opencv2x: Modelling of the V2X cellular sidelink and performance evaluation for aperiodic traffic. *CoRR*, abs/2103.13212, 2021.

[168] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang FlÃ¶tterÃ¶d, Robert Hilbrich, Leonhard LÃ¼cken, Johannes Rummel, Peter Wagner, and Evamarie Wiessner. Microscopic traffic simulation using sumo. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2575–2582, 2018.

[169] Stefan Krauß, Peter Wagner, and Christian Gawron. Metastable states in a microscopic model of traffic flow. *Physical Review E*, 55(5):5597, 1997.

[170] Marco Malinverno, Francesco Raviglione, Claudio Casetti, Carla-Fabiana Chiasserini, Josep Mangues-Bafalluy, and Manuel Requena-Esteso. A multi-stack simulation framework for vehicular applications testing. In *Proceedings of the 10th ACM Symposium on Design and Analysis of Intelligent Vehicular Networks and Applications*, DIVANet '20, pages 17–24, New York, NY, USA, 2020. Association for Computing Machinery.

[171] 3GPP. 5G;Service requirements for enhanced V2X scenarios;. Technical Specification (TS) 22.186, 3rd Generation Partnership Project (3GPP), 11 2020. V. 16.2.0.

[172] Max Robert, Yu Sun, Thomas Goodwin, Hamilton Turner, Jeffrey H. Reed, and Jules White. Software frameworks for sdr. *Proceedings of the IEEE*, 103(3):452–475, 2015.

[173] Cosmin Costache, Octavian Machidon, Adrian Mladin, Florin Sandu, and Razvan Bocu. Software-defined networking of linux containers. In *2014 RoEduNet Conference 13th Edition: Networking in Education and Research Joint Event RENAM 8th Conference*, pages 1–4, 2014.

[174] Aravinthan Gopalasingham, Dalia Georgiana Herculea, Chung Shue Chen, and Laurent Roullet. Virtualization of radio access network by virtual machine and docker: Practice and performance analysis. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 680–685, 2017.

[175] Eryk Schiller, Jesutofunmi Ajayi, Silas Weber, Torsten Braun, and Burkhard Stiller. Toward a live bbu container migration in wireless networks. *IEEE Open Journal of the Communications Society*, 3:301–321, 2022.

[176] Roberto M. Colombo, Aamir Mahmood, Emiliano Sisinni, Paolo Ferrari, and Mikael Gidlund. Low-cost sdr-based tool for evaluating lora satellite communications. In *2022 IEEE International Symposium on Measurements & Networking (M&N)*, pages 1–6, 2022.

[177] E. Saravana Kumar, Raghu Ramamoorthy, Selvaraj Kesavan, T Shobha, Seema Patil, and B Vighneshwari. Comparative study and analysis of cloud container technology. In *2024 11th International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 1681–1686, 2024.

[178] Fabio Busacca, Stefano Mangione, Sergio Palazzo, Francesco Restuccia, and Ilenia Tinnirello. Sdr-lora, an open-source, full-fledged implementation of lora on software-defined-radios: Design and potential exploitation. *Computer Networks*, 241:110194, 3 2024.

[179] Zhenqiang Xu, Shuai Tong, Pengjin Xie, and Jiliang Wang. From demodulation to decoding: Toward complete lora phy understanding and implementation. *ACM Transactions on Sensor Networks*, 18(4):1–27, November 2022.

[180] Laird Connectivity (now Ezurio). RG186 LoRaWAN Gateway: Technical Specification. `https://www.ezurio.com/part/rg186`, 2025. Originally by Laird, now rebranded under Ezurio. Accessed: 2025-02-04.

[181] Carlos Fernandez Hernandez, Oana Iova, and Fabrice Valois. Downlink scheduling in lorawan: Chirpstack vs the things stack. In *2024 IEEE Latin-American Conference on Communications (LATINCOM)*, pages 1–6, 2024.

[182] Dave Conway-Jones, Nick O'Leary, and Julian Rees. Node-red: Low-code programming for event-driven applications. `https://nodered.org/`, 2023. Accessed: 2025-02-04.

[183] Edoardo Di Paolo, Enrico Bassetti, and Angelo Spognardi. Security assessment of common open source mqtt brokers and clients, 2023.

[184] Y Jani. Unified monitoring for microservices: Implementing prometheus and grafana for scalable solutions. *J Artif Intell Mach Learn & Data Sci*, 2(1):848–852, 2024.

[185] Google. cadvisor: Analyzes resource usage and performance characteristics of running containers. `https://github.com/google/cadvisor`, 2024. Accessed: 2025-02-04.

[186] Fabian Graf, Thomas Watteyne, and Michael Villnow. Monitoring performance metrics in low-power wireless systems. *ICT Express*, 10(5):989–1018, 2024.

[187] Mohamed Barhoumi. Quantum-assisted network time synchronisation: A literature review, considering examples and challenges. *Available at SSRN: https://ssrn.com/abstract=5170022*, 2025.

[188] Chrony Project. Chrony: An ntp client and server for time synchronization. `https://chrony-project.org/index.html`, 2025. Accessed on 11 April 2025.

[189] Emiliano Sisinni, Dhiego Fernandes Carvalho, Alessandro Depari, Paolo Bellagente, Alessandra Flammini, Marco Pasetti, Stefano Rinaldi, and Paolo Ferrari. Assessing a methodology for evaluating the latency of ipv6 with schc compression in lorawan deployments. *Sensors*, 23(5), 2023.

[190] Govinda M. G. Bezerra, Nicollas R. de Oliveira, Tadeu N. Ferreira, and Diogo M. F. Mattos. A comprehensive evaluation of software-defined radio performance in virtualized environments for radio access networks. *Annals of Telecommunications*, 79(7-8):523–535, 2024.

[191] Lorenzo Vangelista and Alessandro Cattapan. Extending the lora modulation to add parallel channels and improve the lorawan network performance. In *2021 11th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–5, 2021.

[192] Jiazhao Wang, Wenchao Jiang, Ruofeng Liu, and Shuai Wang. Towards efficient and portable software modulator via neural networks for iot gateways. *IEEE Transactions on Mobile Computing*, 23(12):13866–13881, 2024.

[193] Koen Zandberg, Emmanuel Baccelli, Shenghao Yuan, Frédéric Besson, and Jean-Pierre Talpin. Femto-containers: lightweight virtualization and fault isolation for small software functions on low-power iot microcontrollers. In *Proceedings of the 23rd ACM/IFIP International Middleware Conference*, Middleware '22, pages 161–173, New York, NY, USA, 2022. Association for Computing Machinery.

[194] Anwar Ghammam, Rania Khalsi, Marouane Kessentini, and Foyzul Hassan. Efficient management of containers for software defined vehicles. *ACM Transactions on Software Engineering and Methodology*, 33(8):1–36, 11 2024.

Appendix A

Paper I

Enhancing C-V2X Vehicle Platooning: A Dual-Strategy of Time Synchronisation and Partition-Based Scheduling

Hossein Khalilnasl
Dept. of Information Engineering
University of Brescia, Brescia, Italy
h.khalilnasl@unibs.it

Salvatore Dello Iacono
Dept. of Information Engineering
University of Brescia, Brescia, Italy
salvatore.delloiacono@unibs.it

Paolo Ferrari
Dept. of Information Engineering
University of Brescia, Brescia, Italy
paolo.ferrari@unibs.it

Alessandra Flammini
Dept. of Information Engineering
University of Brescia, Brescia, Italy
alessandra.flammini@unibs.it

Brian McCarthy
Sch. of Computer Science and Inf. Technology
University College Cork, Cork, Ireland
b.mccarthy@cs.ucc.ie

Emiliano Sisinni
Dept. of Information Engineering
University of Brescia, Brescia, Italy
emiliano.sisinni@unibs.it

2024 IEEE International Symposium on Measurements & Networking (M&N) | 979-8-3503-7053-9/24/$31.00 ©2024 IEEE | DOI: 10.1109/MN60932.2024.10615853

***Abstract*—There is clear evidence indicating that the standardized Sensing-Based Semi-Persistent Scheduling scheme in Out-of-coverage possesses specific vulnerabilities. This unsatisfactory performance lies in the absence of coordination in channel scheduling and transmission time. Specifically, the determination to re-assign a resource after the reservation period expires is currently based on a random selection process, without consideration of the actual interference encountered by the utilized resources. In this study, a novel scheme is proposed to markedly improve the random selection method for vehicle positioning.**

***Index Terms*—Cellular V2X, Sidelink, LTE-V2X, NR-V2X, SB-SPS, periodic, CAM, Random Resource Selection, Mode 4, Mode 2, 4G, 5G, Vehicle Platooning.**

I. Introduction

In the dynamic field of vehicular communications, Cellular Vehicle-to-Everything (C-V2X) technology represents a foundational shift toward creating more connected and intelligent transportation systems. C-V2X encompasses a broad spectrum of communication technologies designed to enable vehicles to communicate with each other (V2V), with pedestrians (V2P), with infrastructure (V2I), and with the network (V2N). Within this ecosystem, Long Term Evolution-V2X (LTE-V2X) and its complement, New Radio-V2X (NR-V2X), emerge as pivotal technologies facilitating direct and network-assisted communications crucial for the next generation of vehicular operations.

At the heart of C-V2X's direct communication capabilities are the PC5 (Sidelink) interfaces, enabling Out of Coverage (OoC) communication that does not rely on cellular base stations for data transmission. This feature is crucial for ensuring reliable V2V communication in environments where network coverage is limited or non-existent. In LTE-V2X, this direct mode of communication is implemented through Sidelink Mode 4, while NR-V2X introduces a similar capability with Sidelink Mode 2. Importantly, there is a coexistence between NR-V2X and LTE-V2X according to the 3GPP standard, with the decision that NR-V2X will complement LTE-V2X and not replace it [1].

One of the most promising applications of this direct communication capability is vehicle platooning. This concept involves a group of vehicles traveling close to each other, maintaining synchronized speed and trajectory. Vehicle platooning leverages C-V2X communication to improve road safety, increase transport efficiency, and reduce fuel consumption and emissions.

Central to the efficient operation of C-V2X and, by extension, vehicle platooning, is the Sensing-Based Semi-Persistent Scheduling (SB-SPS) algorithm. This algorithm plays a crucial role in the allocation of communication resources, ensuring that vehicles can reliably exchange information with minimal delay. SB-SPS is designed to optimize the use of available spectrum, taking into account the dynamic nature of vehicular environments.

Given the imperative for platoon members to remain aware of each other's dynamics at all times, including speed, acceleration, and position, prior research indicates that message transmission among platoon members is susceptible to losing data. This loss can lead to the avoidance of data recurrence later and potentially result in temporary lapses in awareness among members, posing a potential safety hazard. This paper aims to enhance the performance of the algorithm for platoon applications through the introduction of a dual-strategy method.

We acknowledge financial support under the National Recovery and Resilience Plan (NRRP), Mission 4, Component 2, Investment 1.1, Call for tender No. 1409 published on 14.9.2022 by the Italian Ministry of University and Research (MUR), funded by the European Union – NextGenerationEU– Project Title Time Sensitive Networking for future enabling technologies: measurement methods and metrological characterization in hybrid wired/wireless scenarios – CUP D53D23016070001 - Grant Assignment Decree No. 1383 adopted on 01.09.2023 by the Italian Ministry of Ministry of University and Research (MUR).

Appendix B

Paper II

Implementing a software defined LoRaWAN node exploiting container-based lightweight virtualization

Hossein Khalilnasl
Department of Information Engineering
University of Brescia, Brescia, Italy
h.khalilnasl@unibs.it

Alessandro Depari
Department of Information Engineering
University of Brescia, Brescia, Italy
alessandro.depari@unibs.it

Paolo Ferrari
Department of Information Engineering
University of Brescia, Brescia, Italy
paolo.ferrari@unibs.it

Alessandra Flammini
Department of Information Engineering
University of Brescia, Brescia, Italy
alessandra.flammini@unibs.it

Massimiliano Gaffurini
Department of Information Engineering
University of Brescia, Brescia, Italy
massimiliano.gaffurini@unibs.it

Emiliano Sisinni
Department of Information Engineering
University of Brescia, Brescia, Italy
emiliano.sisinni@unibs.it

2024 IEEE International Symposium on Measurements & Networking (M&N) | 979-8-3503-7053-9/24/$31.00 ©2024 IEEE | DOI: 10.1109/MN60932.2024.10615549

***Abstract*—The use of microservices based on lightweight virtualization is nowadays a common practice for developing and deploying complex applications. Such an approach has been already profitably adopted for virtualizing functionalities of communication infrastructures. In this paper, the design of a software defined LoRaWAN end node is discussed and the use of Linux Containers is suggested to implement the decomposed functions as isolated microservices. A real-world test bench has been implemented and obtainable performance have been evaluated by means of purposely defined metrics. The feasibility of the proposed approach has been confirmed. In particular, results show that the main limitation is the throughput of the link connecting the host and the Software Defined Radio device (in the order of 40MB/s).**

***Index Terms*—Wireless sensor networks, LPWAN, LoRaWAN, Virtualization, SDR, Containers**

I. Introduction

Many efforts have been carried out in the past to provide an effective abstraction of communication solutions, especially after the advent of the Network Function Virtualization (NFV), Software Define Radio (SDR) and Software Defined Networking (SDN) concepts, that enabled full virtualization of all the components, down to the physical level. Due to the heterogeneity of SDR applications, a component-based model is mandatory in order to clearly define the border (i.e., the interface) and the functions (i.e., the semantic) of each communication mechanism. In this context, any application can be described as an ensemble of transformations applied during transmission and reception of the information of interest. Each transformation can be abstracted into a standalone entity that performs some kind of signal processing or control functionality. Such an approach allows for easier maintenance procedures and enables the reuse of already tested/developed blocks. Additionally, components can have multiple implementations, e.g., depending on the amount of available resources, sharing the same interface and realizing the same functionalities. In turn it means that some kind of virtualization is needed to develop the solution independently from the actual available hardware. It also means that some mechanisms to manage the deployment are required. Another important characteristic is the need for an underlying infrastructure for enabling the communication among components. Generally, a middleware software layer, often addressed as an hypervisor [1], is in charge of providing this infrastructure and a manager supervises the actually available hardware and software resources.

Several purposely designed frameworks have been proposed for addressing these needs, like those described in [2], but there is an increasing attention towards the adoption of traditional virtualization solutions. Notably, considering the need for high performance and real-time, Linux-based containers (LXC) attracted interest as a viable solution for SDR/SDN applications [3]. In particular, this approach has been largely adopted for baseband unit (BBU) of wireless wide area networks, including mobile technologies (4G and 5G) and Low Power Wide Area Networks (LPWANs), as in [4], [5]. In particular, such an approach has been implemented for BBUs in low-complexity solutions targeting Internet of Things (IoT) applications, like LoRaWAN [6]. In a previous work the authors have been the first to successfully adopt this approach to disaggregate and decompose functionalities of a LoRaWAN node, relying on a regular LoRa radio [7].

In this work the LoRaWAN node is fully virtualized, leveraging on a low-cost SDR instead of using a hardwired device. Real-world testbed has been arranged to experimentally evaluate obtainable performance when the node is connected to a regular backend. Docker has been considered as the reference LXC supervisor, exploiting the Compose tool to orchestrate the node functions virtualization. The diverse tasks are implemented by individual containers that are connected

We acknowledge financial support under the National Recovery and Resilience Plan (NRRP), Mission 4, Component 2, Investment 1.1, Call for tender No. 1409 published on 14.9.2022 by the Italian Ministry of University and Research (MUR), funded by the European Union – NextGenerationEU– Project Title Time Sensitive Networking for future enabling technologies: measurement methods and metrological characterization in hybrid wired/wireless scenarios – CUP D53D23016070001 - Grant Assignment Decree No. 1383 adopted on 01.09.2023 by the Italian Ministry of Ministry of University and Research (MUR).

Appendix C

Paper III

electronics

MDPI

Article

On the Use of Containers for LoRaWAN Node Virtualization: Practice and Performance Evaluation

Hossein Khalilnasl *, Paolo Ferrari, Alessandra Flammini and Emiliano Sisinni *

Department of Information Engineering, University of Brescia, 25123 Brescia, Italy; paolo.ferrari@unibs.it (P.F.); alessandra.flammini@unibs.it (A.F.)

* Correspondence: h.khalilnasl@unibs.it (H.K.); emiliano.sisinni@unibs.it (E.S.)

Abstract: This paper investigates the virtualization of LoRaWAN end nodes through Linux containers (LXCs) to improve scalability, flexibility, and resource management. By leveraging lightweight Docker-based virtualization, we break down the core functions of the LoRaWAN node, comprising the application, LoRaWAN, and LoRa layers, into modular containers. In this work, a fully virtualized end node is demonstrated. The obtainable performance is not only compared against the standard approach that leverages a LoRaWAN-compliant module but also against an emulated solution that mimics the desired functionalities purely in software. A controlled, uniform testbed, exploiting the capability of a virtual machine hypervisor to change the way the underlying hardware is abstracted to guest environments, is considered. Key metrics, including resource utilization and latency, are explicitly defined and evaluated. The results underscore the potential of container technologies to transform the deployment and management of communication solutions targeting Internet-of-Things (IoT) scenarios not only for the infrastructure but also for end devices, with implications for future advancements in wireless network virtualization.

Keywords: WSNs; LPWAN; LoRaWAN; virtualization; SDR; containers

Academic Editor: Djuradj Budimir

Received: 21 February 2025
Revised: 4 April 2025
Accepted: 10 April 2025
Published: 12 April 2025

Citation: Khalilnasl, H.; Ferrari, P.; Flammini, A.; Sisinni, E. On the Use of Containers for LoRaWAN Node Virtualization: Practice and Performance Evaluation. *Electronics* **2025**, *14*, 1568. https://doi.org/10.3390/electronics14081568

1. Introduction

In recent years, wired networks have evolved to offer higher and higher flexibility, largely due to the shift from dedicated hardware solutions to software-driven functionalities. This trend has enhanced network performance and scalability, enabling network services to be provisioned, scaled, and optimized with remarkable efficiency. In the past, when dedicated hardware was used for implementing specific network functionalities, the update of an already deployed infrastructure required the introduction of new equipment. Supporting novel user application scenarios and/or protocols, which require disruptive network changes, resulted in a communication infrastructure redesign to comply with the new (standard) specifications. Software-Defined Networking (SDN) and Network Function Virtualization (NFV) technologies, just to mention a few, have played pivotal roles in this transformation by introducing programmability, flexibility, and cost-efficiency into the network infrastructure, for enabling the dynamic reconfiguration of available resources to meet diverse and changing demands [1].

Softwarization and virtualization of networks [2,3] have recently extended into the wireless domain, when challenges such as the efficient use of spectrum and the need for low-latency, high-throughput communication are prominent [4]. The integration of the previously mentioned SDN and NFV solutions into wireless networks has opened new possibilities for flexible network and traffic management. Among the many advancements,