SAPIENZA
UNIVERSITÀ DI ROMA

Investigating Secure and Distributed Control in IoT: Improving BLE Security and Strengthening LoRaWAN with Blockchain

Department of Information Engineering, Electronics and Telecommunications
PhD in Information and Communication Technology (XXXVII cycle)

Pierluigi Locatelli
ID number 1668471

Advisor
Prof.ssa Francesca Cuomo

Academic Year 2024/2025

Thesis defended on 24 Gennaio 2025
in front of a Board of Examiners composed by:

Prof.ssa Daniela De Venuto (chairman)

Prof.ssa Silvia Liberata Ullo

Prof.ssa Cecilia Occhiuzzi

Investigating Secure and Distributed Control in IoT: Improving BLE Security and Strengthening LoRaWAN with Blockchain

Sapienza University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: pierluigi.locatelli@uniroma1.it

Alla mia famiglia, che mi ha dato i mezzi e la serenità per poter scegliere la passione, che mi è stata sempre vicino in questo percorso unico, che mi ha permesso di tirarmi su ogni volta che non mi sono sentito abbastanza.

A Federica "Gran Bel Casino Terremoto Vivente Calcio in Faccia™ ", la luce delle mie giornate, l'unica persona che mi riesce a capire anche quando non riesco a capire me stesso. L'unico tornado che, ogni volta che passa, lascia le cose meglio di quanto non fossero prima.

Ai Drodramussi&Bazzicons, i miei migliori amici, la mia seconda famiglia, anche se ormai siamo diventati una tribù. Purtroppo per voi, i 10 anni di garanzia sono ormai passati, non potete più mandarmi indietro. Grazie per avermi fatto sentire meno solo in questo strano strano mondo.

Al Netlab, compagni di questa strana avventura chiamata dottorato, che mi sono stati vicini durante le mille e una disavventure di questi anni. Tra le tantissime cose, mi avete insegnato che non importa il luogo di nascita. Se abbiamo il cuore nello stesso posto, siamo tutti fratelli, anche se un po' pidorini. Un grazie particolare a Pietro e la professoressa Cuomo, senza i quali oggi non starei scrivendo questa dedica. Non avrei mai pensato di poter anche solo iniziare un percorso simile, figuriamoci finirlo, e buona parte del merito va a voi che avete sempre creduto in me, anche dopo la questione LIM.

A tutti i ragazzi del DIET, per cui dovrò sempre ringraziare l'opendiet, gli unici in grado di reggere un aperitivo a monti di quasi 12 ore, con cui ormai condividiamo una quota dell'elitè di cavour.

Ai miei nonni e a Daniele, che vorrei tanto potessero essere qui a festeggiare con me, spero mi stiate guardando con orgoglio, vi penso sempre.

Abstract

The rapid proliferation of the Internet of Things (IoT) has led to the widespread deployment of low-power wireless communication technologies such as Bluetooth Low Energy (BLE) and LoRaWAN, each designed to meet the connectivity and efficiency demands of IoT devices. BLE has emerged as a key technology for short-range communication, enabling applications such as proximity sensing, wearables, and asset tracking, while LoRaWAN supports long-range communication with low power consumption, ideal for wide-area networks in smart cities and rural areas. However, as the number of connected devices grows, so do the security and privacy concerns associated with these networks. Simultaneously, the advent of edge computing and distributed network paradigms offers potential solutions to some of these challenges, providing enhanced computational power and network decentralization, which are critical for scalable and secure IoT systems.

In BLE networks, Medium Access Control (MAC) address randomization is a key privacy feature, designed to prevent device tracking by periodically changing the device's MAC address. However, by leveraging edge computing, mesh networks of BLE sensors can be deployed to circumvent this feature, enabling large-scale tracking despite randomization.

On the Low-Power Wide-Area Network (LPWAN) side, LoRaWAN typically operates under a centralized architecture, where a Network Server manages key security tasks like authentication and routing. This centralization introduces risks such as single points of failure and insider threats. To address these issues, edge computing can be applied to decentralize LoRaWAN, with edge nodes handling local processes to reduce dependency on the central server. Integrating a permissioned blockchain removes the need for centralized control, ensuring secure, transparent device authentication and key management without relying on a single authority.

This work explores the dual role of edge computing and distributed networks in IoT technologies like BLE and LoRaWAN, examining both the opportunities and risks associated with decentralized approaches. For BLE, the power of edge computing used to circumvent privacy features such as MAC address randomization is investigated. For LoRaWAN, edge computing and permissioned blockchain are proposed as mechanisms to decentralize the network, removing central points of control and improving security against internal and external threats. As IoT continues to expand into various domains, from smart cities to industrial automation, understanding the interplay between edge computing, distributed networks, and low-power communication technologies will be crucial in building scalable, secure, and efficient IoT ecosystems.

Keywords: low-power networks, LoRaWAN, BLE, security, edge computing, decentralization.

Contents

List of Figures

List of Tables

Acronyms

ABP Activation By Personalization

ADR Adaptive Data Rate

AFH Adaptive Frequency Hopping

AoA Angle of Arrival

AoD Angle of Departure

AS Application Server

ATT Attribute Profile

BFT Byzantine-Fault Tolerant

BLE Bluetooth Low Energy

BT Bluetooth

CID Command Identifier

CMAC Cypher-based Message Authentication Code

CSS Chirp Spread Spectrum

CTR Counter-mode encryption

DDoS Distributed Denial-of-Service

DoS Denial-of-Service

EAS Embedded Active Scanning

ECDH Elliptic Curve Diffie-Hellman

ED End Devices

EDR Enhanced Data Rate

FHDR Frame Header

FHSS Frequency Hopping Spread Spectrum

FOpts Frame Options

FPort Frame Port

GAP Generic Access Profile

GATT Generic Attribute Profile

GDPR General Data Protection Regulation

GRU Gated Recurrent Unit

GW Gateway

HCI Host Controller Interface

HF Hyperledger Fabric

IAT Inter-Arrival Time

IEEE Institute of Electrical and Electronics Engineers

IETF Internet Engineering Task Force

IoT Internet of Things

IQR Interquartile Range

ISM Industrial, Scientific, and Medical

JS Join Server

L2CAP Logical Link Control and Adaptation Protocol

LL Link Layer

LoRa Long Range

LPWAN Low-Power Wide-Area Network

LR-WPAN Low-Rate Wireless Personal Area Network

LSTM Long Short-Term Memory

MAC Medium Access Control

MHDR MAC Header

MIC Message Integrity Code

NB-IoT Narrowband IoT

NC Network Controller

NS Network Server

OAS Opportunistic Active Scanning

OSI Open Systems Interconnection

OTAA Over-The-Air Activation

PAN Personal Area Network

PBFT Practical Byzantine-Fault Tolerant

PDR Packet-Delivery Ratio

PDU Protocol Data Unit

PoS Proof-of-Stake

PoW Proof-of-Work

PSR Packet Success Rate

QoS Quality of Service

RF Radio Frequency

RFID Radio-Frequency Identification

RNN Recurrent Neural Network

RPA Resolvable Private Address

RSSI Received Signal Strenght Indicator

SF Spreading Factor

SIG Special Interest Group

SMP Security Manager Protocol

SNR Signal to Noise Ratio

SoC System-on-Chip

SPoF Single Point of Failure

ToA Time on Air

TTN The Things Network

UUID Universally Unique Identifier

WSN Wireless Sensor Network

Chapter 1

Introduction

1.1 Background

1.1.1 Early stages of the Internet of Things

Before the Internet of Things (IoT) became a ubiquitous concept, the connectivity landscape was primarily limited to computers and mobile phones communicating over wired and cellular networks. The roots of IoT can be traced back to the early 1980s when computer scientists at Carnegie Mellon University connected a vending machine to the internet [1]. This allowed them to check the availability of drinks and their temperature remotely, marking one of the first instances of a device connected to the internet to provide real-time data.

In the 1990s, the advancement of internet technologies and the proliferation of mobile devices set the stage for more sophisticated forms of connectivity. The concept of "Smart Devices" emerged, envisioning a world where computation is seamlessly integrated into everyday objects and activities. Kevin Ashton, a British technology pioneer, formally coined the term "Internet of Things" in 1999 while working at Procter & Gamble. He envisioned a system that connects the internet to the physical world via ubiquitous sensors, enabling computers to observe, identify, and understand the world without human intervention.

The early 2000s witnessed significant progress in wireless communication technologies and sensor networks [2]. The development of Radio-Frequency Identification (RFID) and Wireless Sensor Networks (WSNs) played a crucial role in advancing IoT. RFID tags began to be used extensively in supply chain management, retail, and asset tracking, providing a means to identify and track objects automatically. Concurrently, WSN enabled the collection of data from multiple sensor nodes distributed across various locations, facilitating environmental monitoring and military applications.

Despite these advancements, there were limitations in terms of interoperability, power consumption, and communication range. Traditional wireless protocols like Wi-Fi and classic Bluetooth were not optimized for low-power devices, which restricted the scalability of IoT solutions. The need for standardized, energy-efficient communication protocols became increasingly apparent as the number of connected devices continued to grow.

In response to these challenges, the Institute of Electrical and Electronics Engineers (IEEE) introduced the 802.15.4 standard in 2003 [3], which laid the foundation for Low-Rate Wireless Personal Area Network (LR-WPAN). Building upon this standard, protocols like Zigbee and 6LoWPAN were developed to enable low-power, low-data-rate communication suitable for IoT applications. How-

ever, these protocols still faced limitations in terms of interoperability and widespread adoption.

1.1.2 Low Power Networks

A significant breakthrough came in 2010 with the introduction of Bluetooth Low Energy (BLE) as part of the Bluetooth 4.0 specification by the Bluetooth Special Interest Group (SIG) [4]. BLE was specifically designed to provide similar communication ranges to classic Bluetooth but with significantly reduced energy consumption. It utilized a simpler modulation system and introduced features like advertising channels and connectionless communication, allowing devices to operate for years on small batteries. BLE revolutionized the IoT landscape by enabling a new class of devices, such as fitness trackers, smartwatches, medical sensors, and smart home gadgets, that could maintain continuous connectivity without the need for frequent battery replacements or recharging. The widespread adoption of smartphones equipped with BLE further accelerated the integration of BLE devices into consumers' daily lives, acting as hubs for personal area networks and facilitating the seamless exchange of data.

While BLE addressed the need for low-power, short-range communication, there remained a gap for long-range connectivity that was both energy-efficient and cost-effective. Cellular networks were too power-hungry and expensive for many IoT applications, and Wi-Fi's range was limited. This led to the emergence of Low-Power Wide-Area Networks (LPWANs), designed to support devices that need to transmit small amounts of data over long distances with minimal power consumption.

In this context, Long Range (LoRa) technology was introduced by Semtech Corporation around 2013 [5]. LoRa utilized Chirp Spread Spectrum (CSS) modulation, enabling robust, long-range communication even in the presence of interference. Building upon LoRa's physical layer, the LoRa Alliance, a non-profit association formed to standardize LPWANs, released the LoRaWAN specification [6]. LoRaWAN defined the network protocol and system architecture for the network, focusing on scalability, security, and energy efficiency. It allowed for bi-directional communication and supported features like adaptive data rate and end-to-end encryption, making it suitable for large-scale IoT deployments in smart cities, agriculture, industrial monitoring, and environmental sensing.

The introduction of BLE and LoRaWAN addressed the critical challenges of power consumption and communication range, which had previously prevented the widespread adoption of IoT devices. BLE enabled devices to communicate over short distances with minimal energy usage, ideal for personal area networks and wearable technology. In contrast, LoRaWAN provided a solution for long-range connectivity, connecting devices spread over vast areas without the need for significant power resources.

1.1.3 Edge and Cloud Computing

Advancements in various technologies have significantly fueled the growth of the IoT [7], enabling it to scale and become more integrated into different aspects of life and industry. One pivotal development was the transition to IPv6, which addressed the limitations posed by the finite number of IP addresses available under the IPv4 system. As the number of connected devices surged, IPv4's capacity became insufficient, necessitating a new protocol that could accommodate the vast addressing needs of IoT devices. IPv6 provided a virtually limitless pool of IP addresses, ensuring

that each device could have a unique identifier for direct communication, which was crucial for the scalability and interoperability of IoT networks.

Another significant advancement was the development of 6LoWPAN (IPv6 over Low-Power Wireless Personal Area Networks) [8]. This technology enabled the integration of IPv6 in low-power devices, facilitating their seamless incorporation into IP-based networks. By supporting interoperability among different systems, 6LoWPAN played a vital role in expanding the reach and functionality of IoT devices, allowing them to communicate efficiently within existing internet infrastructure.

The rise of cloud computing and big data analytics also played a transformative role in IoT's expansion [9, 10]. The growth of cloud platforms provided the computational power and storage capacity necessary to handle the massive volumes of data generated by IoT devices. Services offered by companies like Amazon Web Services, Microsoft Azure, and Google Cloud allowed for scalable solutions in data analytics, storage, and machine learning. This enabled organizations to process and analyze data more effectively, leading to insights that could drive innovation and improve decision-making processes across various industries. Edge computing emerged as another critical advancement, addressing the need for reduced latency and bandwidth usage in IoT networks. By processing data closer to where it is generated, edge computing enhanced system reliability and enabled real-time responses, which are essential for applications such as autonomous vehicles, industrial automation, and healthcare monitoring systems. This approach not only improved performance but also alleviated the burden on centralized cloud resources, leading to more efficient and resilient networks. Furthermore, innovations in energy efficiency significantly impacted the viability of IoT devices, particularly those reliant on battery power or deployed in remote locations. Developments in energy harvesting technologies allowed devices to draw power from ambient sources such as solar or thermal, extending their operational lifespan. Additionally, the creation of ultra-low-power microcontrollers and communication modules reduced energy consumption, making it feasible for devices to operate over extended periods without the need for frequent maintenance or battery replacements.

The consumer market also played a substantial role in the expansion of IoT. The widespread adoption of smartphones, equipped with various sensors and wireless technologies, positioned them as central hubs within the IoT ecosystem. Smartphones acted as gateways, controlling and communicating with other IoT devices such as smart home appliances, wearables, and health monitors [11]. This integration facilitated greater consumer engagement and convenience, fostering a more connected lifestyle. The introduction of smart home devices further accelerated consumer adoption. Products like smart thermostats, intelligent lighting systems, and voice-controlled assistants brought IoT technology directly into homes, increasing public awareness and acceptance of connected devices. These innovations not only enhanced user experience through automation and personalization but also demonstrated the practical benefits of IoT in everyday life.

1.1.4 IoT Security

The rapid expansion of the IoT has been accompanied by significant security challenges that have evolved [12]. In the early stages of IoT development, security was often an afterthought, with the primary focus on connectivity and functionality. This oversight led to widespread vulnerabilities, as devices were frequently deployed with default passwords, lack of encryption, and minimal

authentication mechanisms. High-profile incidents highlighted these weaknesses; for instance, the 2010 Stuxnet worm targeted industrial control systems, demonstrating how malicious software could disrupt critical infrastructure. Similarly, the 2016 Mirai botnet attack exploited unsecured IoT devices, such as cameras and routers, to launch massive Distributed Denial-of-Service (DDoS) attacks, temporarily bringing down significant portions of the internet [13].

These events underscored the potential for IoT devices to be compromised on a large scale, serving as entry points for cyberattacks that could affect not just individual users but also organizations and national infrastructures. The resource constraints inherent in many IoT devices, such as limited processing power and memory, made implementing traditional security measures challenging. Additionally, the heterogeneity of IoT devices and the lack of standardized security protocols contributed to a fragmented security landscape. In response to these concerns, governments and organizations began to develop regulations and standards aimed at enhancing security and protecting user privacy. The European Union's General Data Protection Regulation (GDPR) [1], for example, imposed strict rules on data protection, mandating transparency and accountability in how personal data is handled. Organizations like the Internet Engineering Task Force (IETF) and industry alliances worked collaboratively to establish standards and best practices that would ensure interoperability and secure communication between devices, fostering a more trustworthy IoT environment.

Despite these efforts, challenges remain due to the evolving nature of cyber threats and the continued proliferation of IoT devices. The history of security issues in the IoT domain illustrates an ongoing struggle to balance the need for low-power, cost-effective devices with the imperative of safeguarding data integrity, privacy, and network resilience. It highlights the necessity for continuous innovation in security solutions tailored to the unique constraints of IoT environments. These advancements collectively propelled the growth of the IoT, overcoming previous limitations related to scalability, interoperability, data processing, and energy efficiency. The integration of IPv6 and 6LoWPAN addressed addressing and network integration challenges, while cloud and edge computing provided the necessary infrastructure for data handling and real-time processing. Innovations in energy efficiency and the emergence of low-power communication protocols like BLE and LoRaWAN made it feasible to deploy IoT devices widely and sustainably. However, these developments also highlighted the importance of addressing security and privacy concerns to ensure the safe and reliable operation of IoT systems. Together, these technological and regulatory advancements set the stage for the continued evolution and integration of IoT into various facets of modern life.

1.2 Novelties

In the next chapters, we will introduce novelties in the field of IoT and Edge Computing, mainly focusing on BLE and LoRaWAN as pivotal technologies in the Low-Power Networks field. We will also discuss security challenges and solutions in these technologies.

Regarding BLE, we will present BLENDER. The BLENDER system introduces several innovative approaches to enhancing BLE device tracking, particularly in overcoming the limitations posed by MAC address randomization. It employs a multi-strategy detection method, which includes both passive listening and the novel *ScanTrigger* mechanism. Passive listening relies on capturing advertising packets, but the *ScanTrigger* goes further by actively provoking devices to

[1] `https://gdpr.eu/what-is-gdpr/`

respond, even when they are not broadcasting advertisements, significantly improving detection in challenging environments. BLENDER also utilizes active fingerprinting through Generic Attribute Profile (GATT) operations, allowing it to collect unique device information like model and manufacturer data to create distinctive fingerprints. This is particularly useful since MAC addresses are very often randomized, so the system can still identify and track devices through these fingerprints. In addition, the JustStore method addresses devices that still use static MAC addresses, such as wearables, offering another layer of detection. Beyond detection, BLENDER facilitates crowd monitoring and mobility tracking, providing insights into movement patterns through the continuous counting of devices and tracking of their unique fingerprints. The integration of LoRaWAN for low-power, long-range data transmission ensures the system's sustainability for extended periods, making it suitable for urban environments.

These innovations enable BLENDER to offer a robust system for monitoring and tracking BLE devices, revealing significant weaknesses in current privacy mechanisms like MAC address randomization and providing new avenues for strengthening BLE security.

On the long-range side, we will present a novel approach to secure and decentralize LoRaWAN networks, called DeLoRaN. The DeLoRaN architecture offers a series of innovative improvements for managing and securing LoRaWAN networks by decentralizing control, integrating blockchain, and incorporating edge computing. These innovations collectively address the limitations found in traditional centralized systems, enhancing both security and performance in IoT environments. DeLoRaN's decentralized network control replaces the single Network Server (NS) with multiple Network Controllers (NCs), distributing key tasks such as packet routing and device authentication. This reduces the risk of a Single Point of Failure (SPoF), increasing reliability and making the network more scalable. The distributed ledger guarantees that data exchanges and authentication events are securely recorded and verified by multiple nodes, removing the need for trust in a central authority. This creates a tamper-proof environment that enhances security in multi-tenant scenarios, where different entities share the same infrastructure. Furthermore, smart contracts automate critical processes like device onboarding and message verification. These self-executing contracts enforce predefined rules, allowing network operations to run smoothly without human intervention, even if certain nodes are compromised. A notable innovation is consensus-based message routing, which requires multiple NCs to agree on the validity of a message before it is processed. This mechanism ensures reliable message delivery and prevents packet manipulation, boosting overall security. DeLoRaN also decouples blockchain processing from device response times, allowing real-time communication with devices while blockchain transactions are handled in the background. This decoupling improves network performance by reducing latency. In addition to blockchain and decentralization, DeLoRaN leverages edge computing to process data closer to the source, improving responsiveness and reducing bandwidth usage. This is particularly effective in large-scale IoT environments where rapid data processing is critical. Lastly, DeLoRaN's modular and scalable architecture allows for the flexible addition of NCs as the network grows, while its efficient resource utilization ensures minimal computational load on any single node. This makes DeLoRaN highly suitable for environments where resources are constrained, offering both scalability and security. In summary, DeLoRaN is a pioneering architecture that enhances the scalability, security, and efficiency of LoRaWAN networks by decentralizing network control, integrating blockchain, and using edge computing to optimize data handling and processing.

1.3 Outline

The rest of the thesis is structured as follows:

In Chap. 2, Bluetooth technology and its low-energy variant are discussed in depth, covering the history, architecture, and security features, along with possible future directions.

Chap. 3 focuses on LoRa and LoRaWAN, providing a detailed breakdown of its data rate, signal quality parameters, end device classes, uplink messages, and other technical aspects of the LoRaWAN protocol. It also addresses LoRaWAN's security features and weaknesses.

Chap. 4 introduces BLE fingerprinting techniques in low-energy nodes and explains how these nodes are detected, enquired, and recognized, touching on experiments and privacy concerns associated with these techniques.

Chap. 5 presents a critical analysis of decentralized low-power wide-area networks (LPWANs), examining centralization issues and how decentralization using blockchain can address some of these problems. It also discusses decentralizing LPWANs using blockchain to enhance security, key management, and load distribution.

Chap. 6 describes the DeLoRaN system, which decentralizes LoRaWAN networks using blockchain. It covers DeLoRaN's architecture, the blockchain's role, security improvements, and specific attacks like Man-In-the-Middle (MITM), Denial-of-Service (DoS), Sinkhole and Blackhole attacks which DeLoRaN seeks to mitigate.

Chap. 7 details the experimental evaluation of DeLoRaN, providing insights into its performance in comparison to centralized LoRaWAN networks and discussing its scalability. It also presents a comparison between DeLoRaN and HyperLoRa and examines real-world data, specifically the LoED dataset.

Chap. 8 explores the decentralized smart adaptive data rate control for LoRaWAN, which improves network efficiency in dynamic environments, and concludes with potential future work and countermeasures for the identified vulnerabilities.

Finally, Chap. 9 concludes the thesis by summarizing the key findings, contributions, and future research directions.

Chapter 2

Bluetooth and Bluetooth Low Energy

2.1 History

Bluetooth (BT), is a wireless communication technology designed for short-range data exchange between devices using short-wavelength Radio Frequency (RF) transmissions in the 2.4 GHz Industrial, Scientific, and Medical (ISM) radio bands. Developed in the late 1990s, BT was intended to eliminate the need for cables connecting personal devices, thereby facilitating seamless wireless communication between mobile phones, computers, and various peripherals. The technology derives its name from the 10th-century Danish king Harald Bluetooth, who united disparate Danish tribes into a single kingdom, symbolizing unification and collaboration among different devices and industries. In 1998, the BT Special Interest Group (SIG) was established by Ericsson, IBM, Intel, Nokia, and Toshiba to oversee the development of the BT standard, promote its adoption, and ensure interoperability among manufacturers. This consortium was crucial in advancing BT technology, fostering widespread acceptance across the electronics industry. Since its inception, BT has evolved through several versions, each introducing enhancements to meet the growing demands for wireless connectivity in an increasingly connected world.

The initial versions, BT 1.0 and 1.0B, released in 1999, faced interoperability issues that hindered widespread adoption [14]. These challenges were addressed in BT 1.1, standardized as Institute of Electrical and Electronics Engineers (IEEE) 802.15.1-2002, which provided a more stable foundation for device communication. This version corrected many early problems, allowing manufacturers to produce devices that could reliably connect and exchange data. The release of BT 2.0 + Enhanced Data Rate (EDR) in 2004 marked a significant technological improvements. This version increased data transfer rates, enhancing performance for data-intensive applications such as file transfers and multimedia streaming. The higher speed facilitated more efficient communication between devices, supporting the growing need for rapid data exchange in personal and professional settings. Subsequent advancements continued this trend, with BT 3.0 + High Speed (HS) introduced in 2009. This version utilized an alternate high-speed transport, typically Wi-Fi, to achieve data rates up to 24 Mbit/s. The integration of Wi-Fi for large data transfers allowed BT to handle bandwidth-intensive tasks while maintaining the simplicity and low power consumption characteristic of BT for smaller data exchanges.

A major milestone in BT technology was the introduction of BT 4.0 in 2010 [4], which brought forth Bluetooth Low Energy (BLE), a protocol optimized for low-power and low-latency applica-

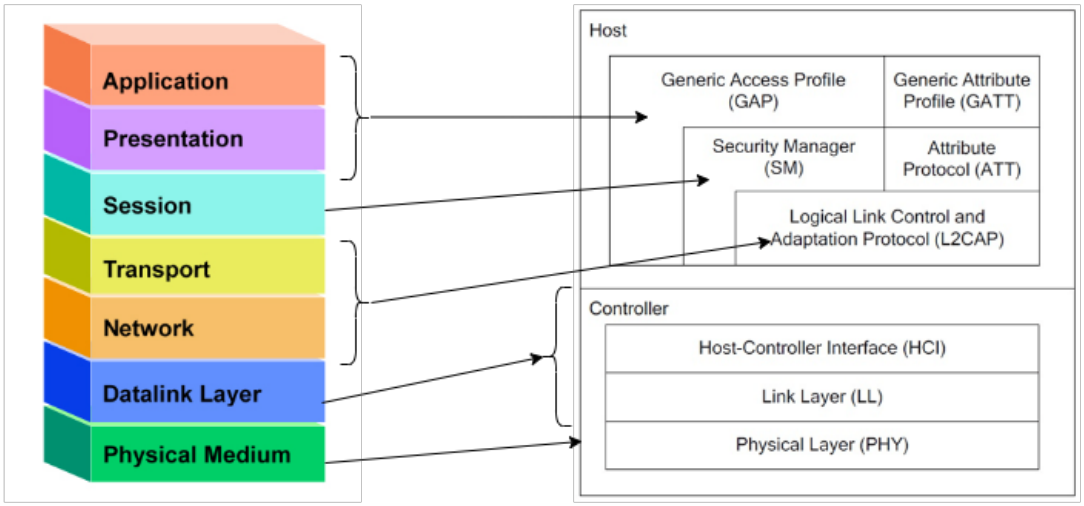


Figure 2.1: BT implements its own layers corresponding to different OSI functions. Key BT protocols include Generic Access Profile (GAP) and Generic Attribute Profile (GATT) in the Host, which manage access and data attributes, while lower layers such as the Link Layer and Physical Layer in the Controller handle radio communication.

tions. BLE enabled devices such as fitness trackers, medical sensors, and smartwatches to operate for extended periods on small batteries, revolutionizing the wearable technology market. This development significantly expanded the scope of BT, making it fundamental to the developing Internet of Things (IoT) landscape by supporting devices that require minimal energy consumption without sacrificing connectivity. Further advancements came with the BT 5 series, beginning with BT 5.0 in 2016 [15]. These versions enhanced BLE features by offering increased range, up to 240 meters under ideal conditions, higher speeds and larger broadcast messaging capacity.

The latest addition to the BT standard is Bluetooth Mesh Networking [16], officially introduced by the BT SIG in July 2017. Its development was driven by the increasing demand for robust, scalable networks capable of supporting the developing IoT. Before mesh networking, BT technology was primarily used for point-to-point or small-scale star topologies within Personal Area Networks (PANs), which were insufficient for applications requiring communication across hundreds or thousands of devices. This new standard enhanced network resilience and scalability, making it suitable for complex applications and industrial IoT systems. The adoption of BT Mesh has been facilitated by the widespread availability of BLE hardware and the familiarity of developers with BT technology. Ongoing enhancements to the mesh specifications continue to improve performance, scalability, and ease of deployment, ensuring that BT remains a key enabler in the evolving IoT ecosystem.

2.2 Architecture

The architecture of BT, shown in Fig. 2.1, is divided into two main components: the Controller and the Host subsystems, together with a layered approach similar to the Open Systems Interconnection (OSI) model.

The Controller subsystem encompasses the Physical Layer (PHY) and the Link Layer (LL). The PHY operates in the globally available 2.4 GHz Industrial, Scientific, and Medical (ISM), utilizing Gaussian Frequency Shift Keying (GFSK). This enables reliable, short-range communication with

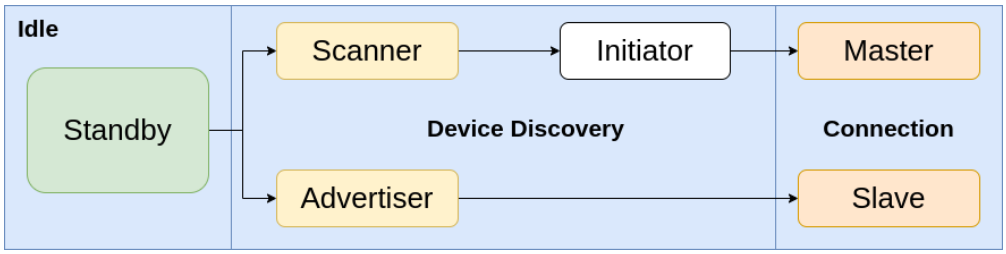


Figure 2.2: BT goes through different states during a connection, depending on the role .

up to 3 Mbps data rates. Bluetooth Classic divides the 2.4 GHz spectrum into 79 channels, each 1 MHz wide, and uses Frequency Hopping Spread Spectrum (FHSS) to switch channels up to 1600 times per second. This reduces interference and improves security by minimizing the risk of signal eavesdropping.

In contrast, BLE operates within the same 2.4 GHz band but uses only 40 channels, each 2 MHz wide. BLE's reduced channel set (Fig. 2.3) includes three advertising channels, used for broadcasting connection requests and beaconing, and 37 data channels. This more focused use of the spectrum helps minimize collisions with other wireless technologies like Wi-Fi, which operate in the same band. Additionally, BLE's physical layer is optimized for low-power, intermittent data transmission, with data rates up to 2 Mbps while maintaining high efficiency in energy consumption.

The LL manages the lower-level protocols and procedures that establish and maintain wireless connections between devices. It orchestrates critical functions such as device advertising, scanning for other devices, initiating connections, and handling data channel communications. There are six distinct states that a BLE device can be in, shown in Fig. 2.2, depending on its role and communication requirements:

- **Standby**: This is the idle state where the device is not transmitting or receiving any data and is not connected to another device.
- **Advertiser**: In this state, the device periodically broadcasts advertisement packets to make itself known to other nearby devices. This is an essential step in initiating communication;
- **Scanner**: The device actively listens for advertisements from other devices. It is used when the device is searching for other devices broadcasting their presence;
- **Initiator**: This state is where the device attempts to initiate a connection with another device, typically after detecting its advertisement;
- **Master**: Once connected, the device can act as a master, controlling communication with another device by managing data exchanges;
- **Slave**: In contrast, the slave state indicates the device is connected to a master device and responds to its requests and commands;

When it comes to the advertising feature in BLE, there are four distinct types of advertisements that devices can use to broadcast their presence and capabilities:

- **Connectable Undirected**: This is the most common advertisement type where any scanning device can initiate a connection with the advertiser. The advertiser is open to connecting with any available device:

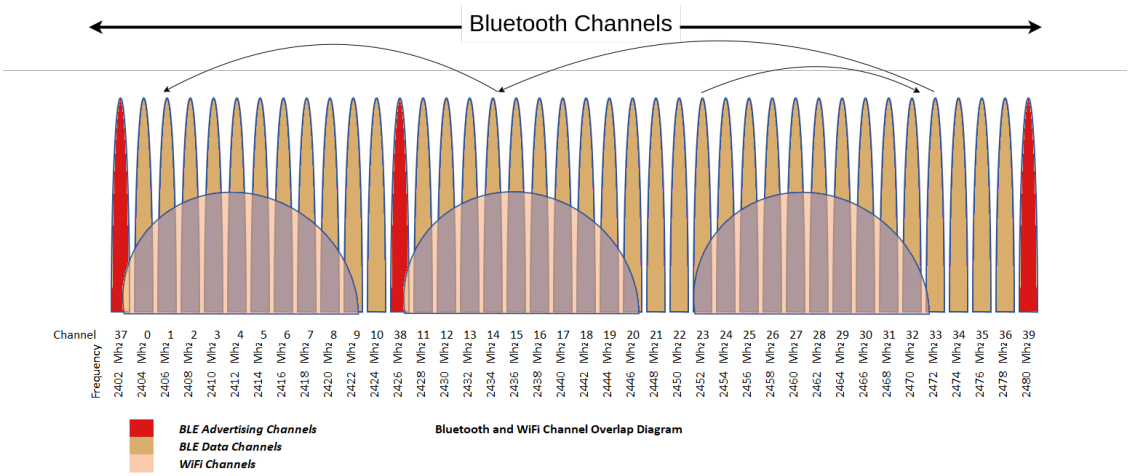


Figure 2.3: This diagram shows the overlap between BLE and Wi-Fi channels in the 2.4 GHz ISM band. BLE uses frequency hopping across its Data Channels (in pink) to minimize interference with Wi-Fi channels (in beige). The BLE Advertising Channels (in red) are spaced strategically to avoid the most congested parts of the spectrum. This frequency-hopping mechanism ensures reliable communication even in environments with overlapping Wi-Fi and Bluetooth traffic.

- **Connectable Directed**: In this mode, the advertiser broadcasts its availability for connection, but only a specific, pre-designated device is allowed to initiate a connection;
- **Non-connectable Undirected**: This type of advertisement is used when the device is broadcasting information but is not open to any connection. It is primarily used for broadcasting data without the need for device pairing;
- **Discoverable Undirected**: In this mode, the advertiser broadcasts its presence and information, allowing any scanner device to request additional details. However, the advertiser will not accept connection requests in this mode;

The LL manages device behaviour in these various roles and the transition between those roles. It also handles low-level security features, including packet encryption and authentication, by interfacing with the Security Manager Protocol (SMP) in the higher Host subsystem. The LL is responsible for link setup and control between BT devices, including authentication, encryption, and power management. This layer establishes and maintains the connection, negotiating parameters and managing the security aspects of the communication. The Host Controller Interface (HCI) provides a standardized interface for accessing the hardware capabilities of the BT module, facilitating communication between the host system, such as a computer or smartphone, and the BT controller. This separation allows for modular design and easier integration of BT into various devices.

Moving to the Host subsystem, it comprises higher-level protocols and profiles that facilitate data exchange and application-specific functionalities. The Logical Link Control and Adaptation Protocol (L2CAP) serves as a multiplexing layer, allowing multiple higher-level protocols and applications to share the lower-level Link Layer connections. L2CAP provides essential services such as segmentation and reassembly of data packets, protocol multiplexing, and Quality of Service (QoS) management, which are vital for maintaining efficient and organized communication channels. Central to BLE's data management is the Attribute Profile (ATT), which defines a client-server

architecture where the server maintains a set of attributes accessible to clients. These attributes represent data structures that include services, characteristics, and descriptors, each identified by a unique Universally Unique Identifier (UUID). The GATT builds upon ATT by specifying how attributes are grouped into services and how they can be discovered, read, written, and notified. GATT outlines procedures for service discovery, characteristic manipulation, and subscription to value change notifications or indications, enabling flexible and efficient data exchange tailored to specific application requirements. The GAP manages all device discovery, connection establishment, and link management aspects. It defines devices' roles, Peripheral, Central, Broadcaster, and Observer, each with specific behaviours and responsibilities. Peripherals, typically resource-constrained devices like sensors, advertise their presence and accept connections initiated by Centrals, which are usually more capable devices like smartphones or tablets. GAP ensures that devices can discover each other, establish secure connections, and manage ongoing communications effectively.

2.3 Network Topology

In terms of network topology, BT devices form networks called PANs, which are based on a master-slave architecture. A basic BT network, known as a piconet, consists of one master device and up to seven active slave devices. The master controls the communication link and timing, coordinating the exchange of data within the network. Multiple interconnected piconets can form scatternets, where devices participate in more than one piconet, acting as a bridge and facilitating more complex networking arrangements. This flexibility allows BT to support a variety of networking scenarios, from simple point-to-point connections to more intricate network structures.

To support all the different network topologies, BT supports various communication models, as in Fig. 2.4, tailored to different needs. It operates primarily in connectionless (broadcast) or connection-oriented (paired) modes:

- **Connectionless Communication**: Supports one-to-many (1:N) interactions, typically involving a broadcaster, which regularly transmits data but does not accept incoming connections (e.g., beacons), and an observer, which passively listens for broadcasted messages without establishing direct links.
- **Connection-Oriented Communication**: Designed for one-to-one (1:1) interactions, involving roles such as Central and Peripheral.
- **Mixed communication**: Supports many-to-many (N:M) interactions, commonly used in BLE mesh networking to form local or personal area networks. In all models, the connection is immediately terminated when the transmission is over, optimizing energy consumption.

BLE Mesh

To address the need for large-scale device networks, BT Mesh extends BT's capabilities by utilizing a many-to-many topology, allowing devices to relay messages across the network. This mesh networking enables communication over greater distances and around obstacles by passing messages through intermediate devices, effectively expanding the coverage area. BT Mesh is particularly suited for building automation, smart lighting, and industrial IoT applications where reliable communication

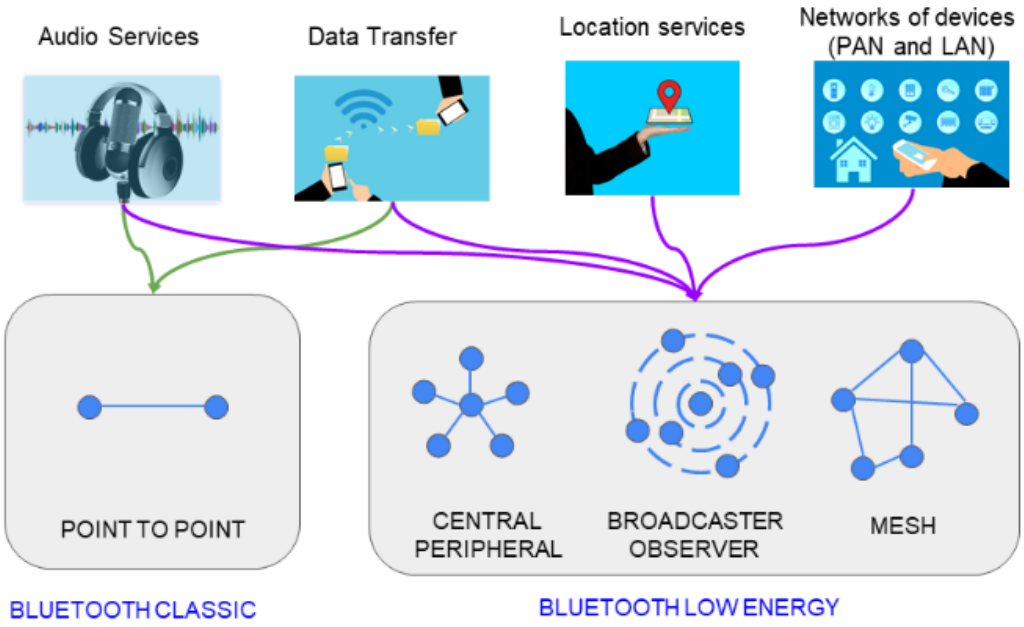


Figure 2.4: BT Classic supports point-to-point connections, primarily used for audio services and high-throughput data transfer. BLE, on the other hand, offers a broader range of communication models, including Central-Peripheral for data transfer, Broadcaster-Observer for location services, and Mesh networking for large-scale device networks, such as personal and local area networks.

across numerous devices is required. Unlike the traditional BT point-to-point or star topologies, BT Mesh allows devices to communicate directly or through intermediary nodes, forming a self-healing network that can reroute messages if any node fails. This topology significantly enhances network resilience, as nodes can relay messages to extend the communication range beyond single-hop connections. In BT Mesh, message transmission is based on a managed flooding mechanism. Messages are broadcasted and relayed through multiple nodes until they reach their intended destination. To optimize this process, techniques such as Time-to-Live (TTL) counters and message caching are used to control the flow of traffic and prevent unnecessary retransmissions, ensuring the network operates efficiently. Despite the initial impression that flooding might be inefficient, these strategies make BT Mesh well-suited for reliable, large-scale communication [17].

2.4 Bluetooth Security

Security is integral to BT, with mechanisms designed to protect against eavesdropping, unauthorized access, and other threats. The pairing process involves devices exchanging link keys and establishing trust relationships. Various pairing methods exist, each offering different levels of security and user interaction. These methods balance ease of use with security requirements, allowing users to connect devices conveniently while maintaining protection against unauthorized access.

To ensure robust communication in the crowded 2.4 GHz band, BT employs several techniques to mitigate interference and maintain connection quality. One such method is FHSS, where devices rapidly switch frequencies, 1600 hops per second, across 79 channels in a pseudo-random sequence. This approach reduces the likelihood of interference from other devices operating in the same fre-

quency band, such as Wi-Fi networks or microwave ovens, and also enhances security by making it more difficult for unauthorized parties to intercept communications. Additionally, Adaptive Frequency Hopping (AFH), introduced in BT 1.2, enhances FHSS by detecting occupied channels and avoiding them. By dynamically adjusting the hopping sequence to exclude frequencies with high levels of interference, AFH minimizes communication disruptions and improves coexistence with other wireless technologies. This adaptive approach allows BT devices to operate more efficiently in environments with heavy RF activity.

Authentication and encryption ensure that devices are communicating with trusted partners, using challenge-response protocols and encryption algorithms to protect data. Classic BT uses stream cyphers like E0 for encryption, while BLE employs AES-CCM for stronger security. BT defines multiple security modes and levels, allowing flexibility based on application requirements. Secure Simple Pairing (SSP), introduced in BT 2.1 + EDR, improves security and usability during the pairing process by using Elliptic Curve Diffie-Hellman (ECDH) for key exchange, providing robust protection against passive and active attacks.

2.5 Bluetooth Low Energy Security

Security in BLE is managed by the SMP module, which operates in the Host subsystem to establish secure communication between devices. The SMP facilitates pairing, during which devices exchange security capabilities and generate shared secret keys using algorithms like ECDH. This ensures mutual authentication, creating a trusted link that protects against eavesdropping and man-in-the-middle attacks. Additionally, the SMP distributes keys for encryption, authentication, and data signing, enabling secure reconnection and device identification.

A crucial privacy feature managed by the SMP is the use of Resolvable Private Address (RPA), which change periodically to prevent tracking. Fixed addresses can be monitored by attackers, but RPA mitigates this risk by using temporary addresses that change typically every 15 minutes. RPA is generated using a cryptographic function based on a secret Identity Resolving Key (IRK), shared only with trusted devices during pairing. When a device broadcasts an RPA, trusted devices with the IRK can resolve it, allowing seamless, secure communication while keeping the device's identity hidden from unauthorized scanners.

In BT Mesh, security is enforced using multiple layers of encryption and authentication, with network and application keys protecting messages. This multi-layered approach ensures the integrity of communication and prevents unauthorized access, which is vital for controlling critical systems like lighting and access control in mesh networks.

2.6 Future Directions

The future directions of BT technology continue to focus on innovation and enhancement to meet emerging needs. The BT SIG is working on enhanced audio capabilities, such as LE Audio, for improved audio quality and energy efficiency. LE Audio aims to deliver high-quality sound while reducing power consumption, making it ideal for hearing aids and wireless earbuds.

Advancements in location services are also a focus, improving accuracy for indoor positioning through techniques like Angle of Arrival (AoA) and Angle of Departure (AoD). These methods

enable devices to determine the direction of a signal, allowing for precise positioning and tracking. Applications include asset tracking in warehouses, wayfinding in large facilities like airports and shopping malls, and enhanced user experiences in augmented reality.

Ongoing efforts aim to optimize energy efficiency further, reducing power consumption and extending the battery life of BT devices. This is critical for devices that are expected to operate for years without maintenance, such as environmental sensors and industrial monitoring equipment. Additionally, ensuring interoperability and coexistence remains a priority, with efforts to make BT operate seamlessly alongside other wireless technologies, minimizing interference and enhancing user experience.

BT has evolved into a versatile and ubiquitous wireless technology, integral to modern connectivity across consumer electronics, automotive systems, healthcare, and industrial applications. Its continuous development, particularly with the introduction of BLE and mesh networking, addresses the diverse needs of the expanding IoT landscape. Understanding the fundamentals of BT is crucial for leveraging its capabilities and addressing security and performance challenges inherent in wireless communication systems. As BT continues to adapt and innovate, it remains a key enabler of the connected world, facilitating the seamless interaction of devices that enrich daily life and drive technological progress.

Chapter 3

LoRa and LoRaWAN

3.1 LoRa

Long Range (LoRa) is a proprietary Low-Power Wide-Area Network (LPWAN) modulation technique [18]. It allows smart objects to communicate over long distances at a very low energy cost, enabling devices powered by simple batteries that last for years. It is based on spread spectrum modulation techniques derived from Chirp Spread Spectrum (CSS) technology, where a chirp is a high-frequency signal that increases or decreases rapidly over time. The LoRa technology covers only the physical layer of the transmission, while other protocols like LoRaWAN cover the upper layers, managing message transport, security, and performing operations to optimize the radio space. LoRa operates on various frequencies, like 433 MHz and 868 MHz in Europe or 915 MHz in North America, and at different data rates called Spreading Factor (SF).

3.1.1 Data rate: Spreading factor, Bandwidth and Time-On-Air

SF in LoRa technology represents the number of bits transmitted per symbol, with possible values ranging from 7 to 12. A lower SF allows more bits per symbol, resulting in more chirps per second and higher data rates. However, this comes at the cost of reduced sensitivity, leading to a shorter transmission range (an example of LoRa's range is shown in Fig. 3.1). Conversely, a higher SF reduces the number of chirps per second but increases sensitivity, thereby extending the transmission range. For each increment in the spreading factor, the chirp sweep rate is halved, effectively halving the data transmission rate, as illustrated in Table 3.1. The bandwidth also influences the data rate; doubling the bandwidth doubles the bit rate for a fixed spreading factor. LoRa supports three bandwidths: 125kHz, 250kHz, and 500kHz. Using a higher SF requires more transmission time, known as airtime or Time on Air (ToA), which in turn increases the energy consumption of the device. For example, to transmit 20 bytes at a bandwidth of 125kHz with SF values ranging from 7 to 12, the ToA values are approximately 71.9ms, 133.6ms, 246.8ms, 452.6ms, 987.1ms, and 1810.4ms, respectively. This demonstrates how the ToA nearly doubles with each increase in SF, due to the halving of the chirp sweep rate.

For an Internet of Things (IoT) device, selecting a lower SF is generally preferable, as it minimizes airtime and energy consumption. However, LoRa does not provide a mechanism to optimize transmissions for the best trade-off between data rate and transmission range. Moreover, the LoRa standard includes regional regulation on the amount of time a device can be actively transmitting.

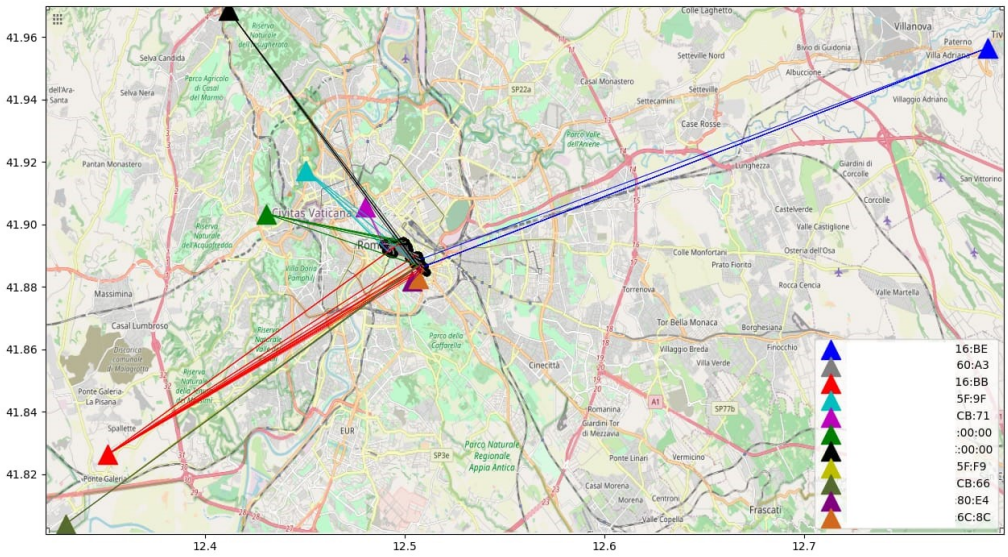


Figure 3.1: We sent packets in different urban locations. In the figure are reported the position in which we sent a packet and the approximate position of the gateway receiving that packet. The packets which travelled the longer distance are reported with the blue colour, covering a distance of more than 27km. All the tests were performed with SF=12.

Table 3.1: In the table is reported the raw LoRa bit rate as a function of the Spreading Factor. A bandwidth of 125kHz and a coding rate of CR=4/5 were used for the calculations, the most commonly used in the wild.

Spreading Factor	Bitrate (kbps)
SF7	5.468
SF8	3.157
SF9	1.761
SF10	0.978
SF11	0.544
SF12	0.219

This parameter, called duty cycle, is the proportion of time during which a device actively communicates with the network and is usually expressed as a percentage. In Europe, the standard imposes that the duty cycle must be around 0.1% (at most 1.0%) per day depending on the channel used. For example, if we have a duty cycle of 0.1% and a ToA of 500ms, we can calculate the maximum number of messages the device can send in a day. A 0.1% duty cycle means the device can only transmit for 0.1% of each hour, which is equivalent to 3.6 seconds per hour.

Given a ToA of 500ms (0.5 seconds), each message takes 0.5 seconds to send. Therefore, the device can send a maximum of:

$$\frac{3.6 \text{ seconds per hour}}{0.5 \text{ seconds per message}} = 7.2 \text{ messages per hour}$$

In a 24-hour period, this results in:

$$7.2 \text{ messages/hour} \times 24 \text{ hours} \approx 173 \text{ messages/day}$$

Thus, under a 0.1% duty cycle and a ToA of 500ms, the device can send up to 173 messages per day.

3.1.2 Signal Quality Parameters: RSSI and SNR

The Received Signal Strenght Indicator (RSSI) represents the power of a received signal measured in decibel-milliwatts (dBm). It serves as an indicator of how well a receiver can detect the signal sent from a device. In LoRa communication, RSSI values typically range between -120dBm and 0dBm. A value of -120dBm is considered the minimum threshold for successful communication without the signal being discarded. Values around -30dBm indicate optimal signal strength, often corresponding to scenarios where the device is only a few meters away from the receiver.

The Signal to Noise Ratio (SNR) is the ratio between the power of the received signal and the power level of the noise floor. The noise floor comprises unwanted interfering signals that can corrupt the transmitted signal, potentially leading to retransmissions from the sender. In LoRa technology, typical SNR values range from -20dB to +10 dB. An SNR greater than 0dB means the received signal operates above the noise floor, indicating low interference. Conversely, an SNR less than 0dB implies that the signal operates below the noise floor, which can result in increased signal corruption. Although the noise floor is generally the physical limit of sensitivity, LoRa can demodulate signals even when the SNR is between -7.5 dB and -20 dB.

Since both RSSI and SNR are related to the quality of the transmission, it is common for a signal with a good RSSI value to also exhibit a good SNR value.

3.2 LoRaWAN

To address the limitations of LoRa, upper-layer protocols have been developed to provide Medium Access Control (MAC), networking, and security functionalities on top of it. In recent years, the most widely adopted protocol is LoRaWAN, which enables the creation of a Wide Area Network based on LoRa [19, 20]. The first version of the LoRaWAN standard was released in January 2015, and since then, minor updates have been published approximately once a year to fix bugs and resolve various security issues. A significant change occurred in October 2017 with the release of LoRaWAN 1.1 [21], introducing major security enhancements and stricter rules regarding the integrity and confidentiality of communications.

Despite the release of LoRaWAN 1.1, virtually every LoRaWAN device continued to operate on version 1.0.X. To avoid forcing customers to purchase new devices capable of running the updated protocol while still providing a more secure and stable version, updates for the 1.0.X series continued, culminating in the release of version 1.0.4 in October 2020 [6]. Unless otherwise specified, when we refer to LoRaWAN, we implicitly mean version 1.1.

A LoRaWAN network employs a star-of-stars topology (Fig. 3.2). At the very edge of the network we have End Devicess (EDs), the leaves of the architecture and the real IoT devices, communicating over long distances using LoRa; then we have the gateways, which receive LoRa frames and forward them over the internet to the centralized Network Server (NS); the NS acts as

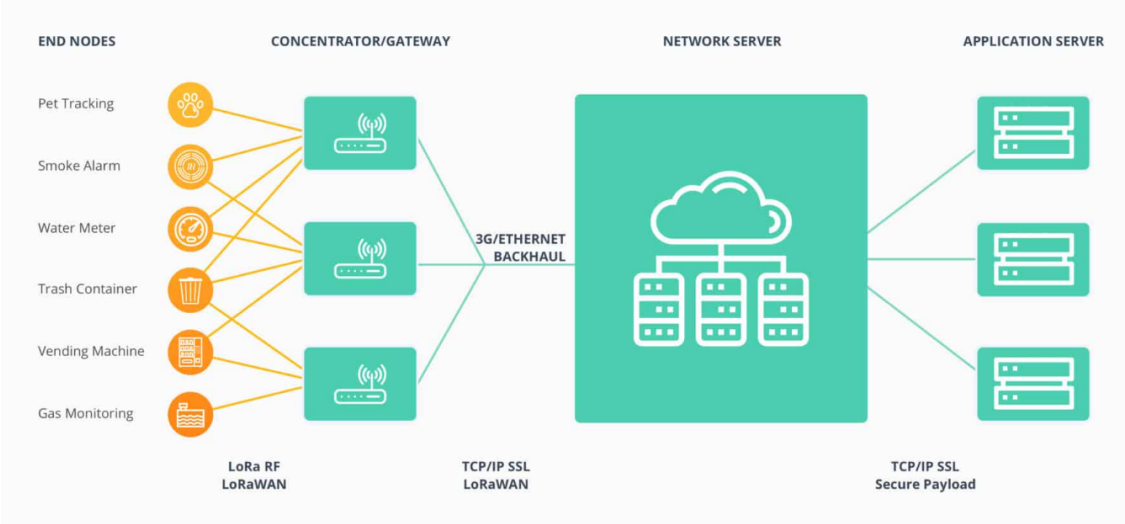


Figure 3.2: LoRaWAN has a star-of-stars topology. Each leaf node, an ED, can directly reach one or more sink nodes called Gateways (GWs). These gateways forward transmissions to a centralized NS. After extracting the payloads from the packets, the NS dispatches them to the appropriate AS.

the head of the network, ensuring the identity of the device and the integrity of the transmission; finally, we have the Application Servers (ASs), which collect the payloads sent by the end devices and act accordingly.

3.2.1 End Device Classes and Activation

LoRaWAN End Devices can be divided into three main categories (Fig. 3.3):

- **Class A** (Aloha) devices support bi-directional communication, primarily sending data to the server with rare downlink messages. Uplink messages can be sent anytime, while downlink messages are scheduled after an uplink transmission. The device opens two receive windows at specified times, "RX1 Delay" and "RX2 Delay" (typically RX1 + 1s). If a downlink is received during RX1, RX2 will not open. If no downlink is received, the next opportunity will occur after the next uplink. Being the most energy and resource-efficient is the most widely used class for EDs;
- **Class B** (Beacon) builds on Class A by adding scheduled receive windows. The ED synchronizes its clock with periodic beacons broadcast by the network. Based on this timing, EDs open "ping slots" at set intervals, allowing the network to initiate downlink;
- **Class C** (Continuous) extends Class B by keeping the receive window open continuously, except when transmitting. This setup supports low-latency communication but requires much more power, making Class C suitable only for devices with continuous power sources, limiting their use in energy-constrained IoT applications;

Alongside the device classes, which define the behaviour of uplinks and downlinks, another important factor divides LoRaWAN devices into two categories: the method of activation of the ED. There are two types of activation: Over-The-Air Activation (OTAA) and Activation By Personalization (ABP).

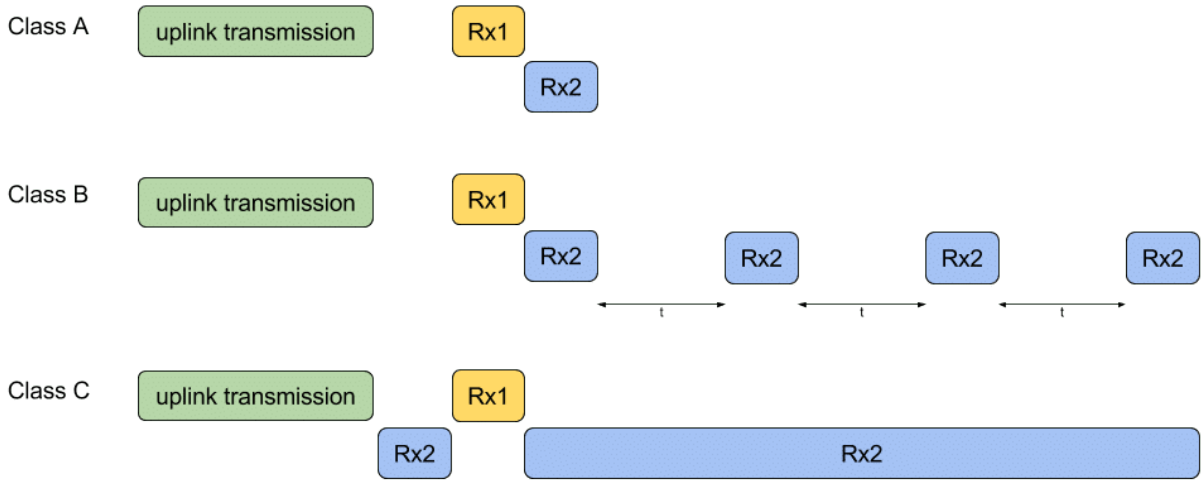


Figure 3.3: Approaches used by different LoRaWAN EDs classes to handle downlink communication. In the figure, each plot represents a timeline. Class A (Aloha) has only two well-defined downlink windows which are opened only after an uplink. Class B (Beacon) extends class A devices with a downlink window that is opened at constant intervals of time. Class C (Continuous) has always a downlink window opened, except during uplink communications.

ABP activation is very simple, as the device is already fully configured and does not need to exchange any information with the NS, which knows *a priori* all the encryption session keys (an ABP device has only one session that lasts its lifetime) (Fig. 3.4), and the DevAddress, which functions like an IP address in the LoRaWAN protocol. This information is static and cannot be changed, as it is "hardcoded" into the Electrically Erasable Programmable Read Only Memory (EEPROM) of the device. Therefore, the buyer simply needs to input this information into the NS, and the ED becomes fully operational and ready to send messages over the network. However, the use of ABP devices is discouraged by the LoRaWAN specification due to inherent security weaknesses. The static DevAddress and encryption keys, if compromised, allow an attacker to decrypt all subsequent transmissions.

In contrast, OTAA activation requires EDs to follow a join procedure before they can exchange messages with the Network Server. An ED may undergo a new join procedure for various reasons, primarily to refresh its session information or during its initial connection. Unlike ABP, OTAA activation requires the ED to be personalized with specific information before initiating the join procedure: a DevEUI, which acts like a MAC address in LoRaWAN; a JoinEUI, which identifies the join server that will handle the join request; and the NwkKey and AppKey, which are root keys used to derive the network session key (for encrypting communications between the ED and the NS) and the application session key (for encrypting the payload sent to the application server). The entire key derivation scheme for LoRaWAN 1.1 is depicted in Fig. 3.4, while the scheme for LoRaWAN 1.0.X is shown in Fig. 3.5.

3.2.2 Uplink messages

Uplink messages, messages sent by the ED to the AS, can be sent at any moment and are the most frequent type of communication in a LoRaWAN network. End Devices are mostly just sensors or

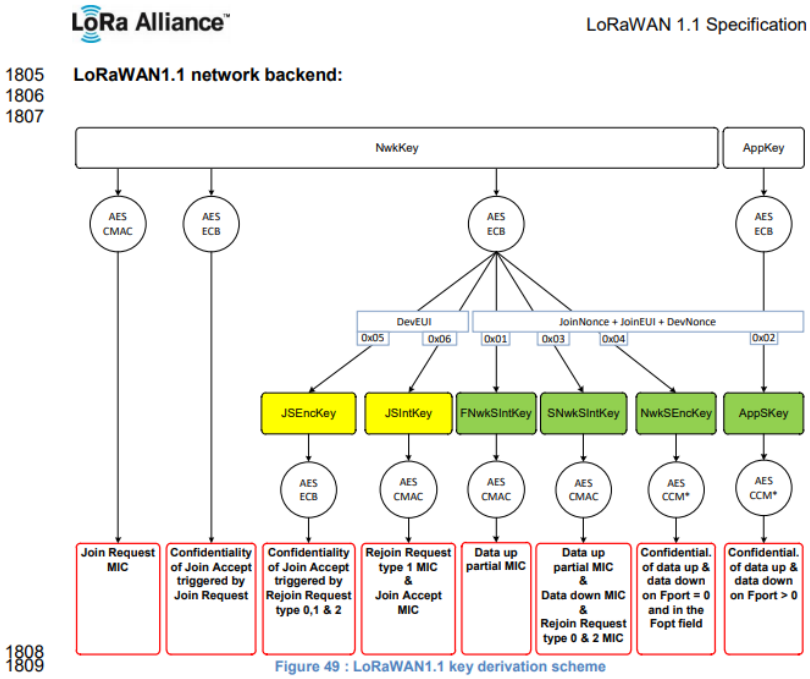


Figure 3.4: The key-derivation scheme used by a LoRaWAN 1.1 ED and NS to calculate the different session keys after a successful join procedure, complete with the goal of each key.

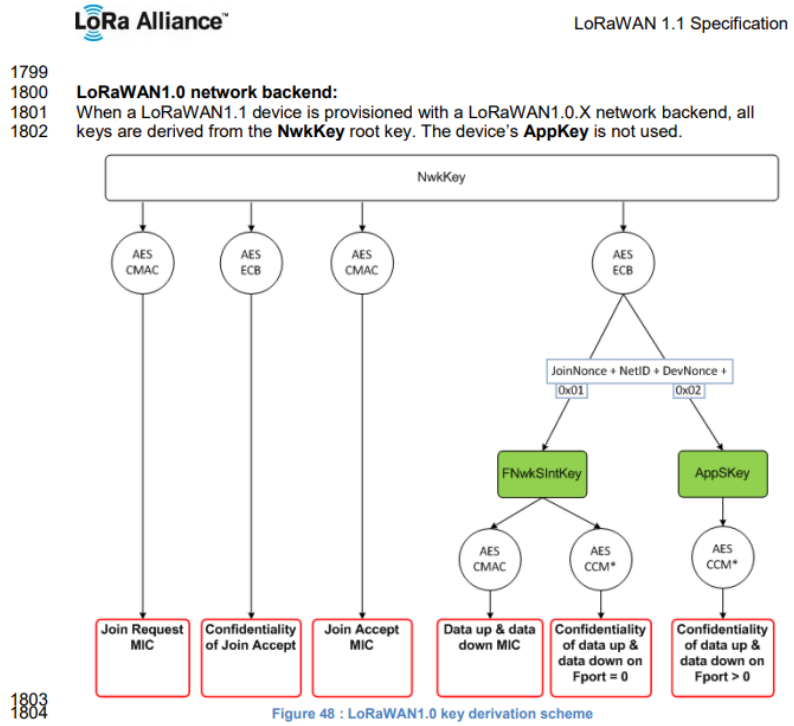


Figure 3.5: The key-derivation scheme used by a LoRaWAN 1.0 ED and NS to calculate the different session keys after a successful join procedure.

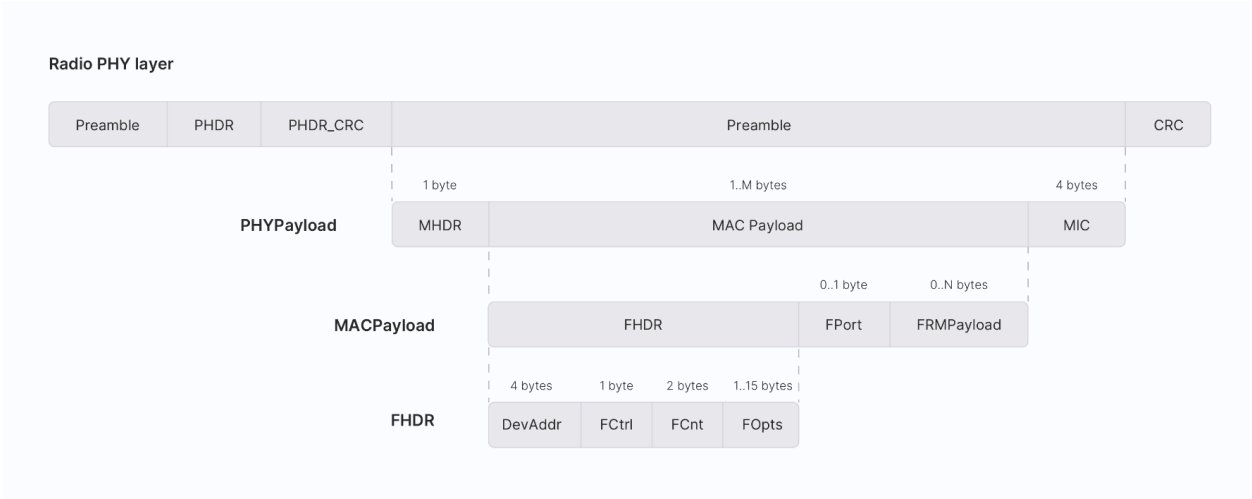


Figure 3.6: The structure of a LoRaWAN frame. Being built on top of LoRa, the physical layer has the structure of a LoRa frame. Its PHYPayload is then divided into MAC Header (MHDR), MAC payload, and a Message Integrity Code (MIC) to check the integrity of the whole packet. The MAC Payload contains application data, which again is divided into Frame Header (FHDR), FPort (used to distinguish different applications), and its payload

embedded systems that are powered by batteries, or even solar panels, so it is imperative to be as energy-saver as possible. Because of this, the average uplink rate is just a few bytes sent once per day, but, in any case, it can not exceed the duty cycle imposed by the standard or local regulation. When an ED sends an uplink, because of its broadcast nature, it is common for the message to be received by many different gateways, which will then forward each one the message to the network server. The NS has to find a way to deal with multiple copies of the same message and forward the message to the AS only once. This step is called *Deduplication* and it is a necessary step in LoRaWAN, despite being implementation dependant. At the end, everything boils down to this *deduplication window*, during which the NS will accept and aggregate every duplicate it receives and, after the expiration, it chooses the the packet with the best value of SNR and/or RSSI, or a random packet between the ones with a value of RSSI and/or SNR over a certain threshold. Every duplicate packet coming after the end of the window will be automatically discarded by the NS without even reading any of these values. In this way the NS will always upload only one packet for every group of duplicated messages received and will keep track in its own database of the "best gateway" for every device, which is the gateway that forwarded the uplink with best values for SNR/RSSI for each device in the network, as this information will be used later to retrieve the correct gateway to schedule a downlink message.

3.2.3 Control-level MAC Messages

In LoRaWAN, in addition to "application-level" communication between the AS and the ED, there are messages that the NS must handle differently at the MAC-layer level. These messages are identified by the first bytes of the LoRaWAN packet in the so-called MHDR, specifically in its MType field, and have different meanings as described in Table 3.3. MAC commands play a crucial role in network control and management, enabling communication between the NS and the ED to configure device parameters, manage network settings, and perform other control functions. These commands are embedded within the payload of LoRaWAN frames and are carried in the Frame

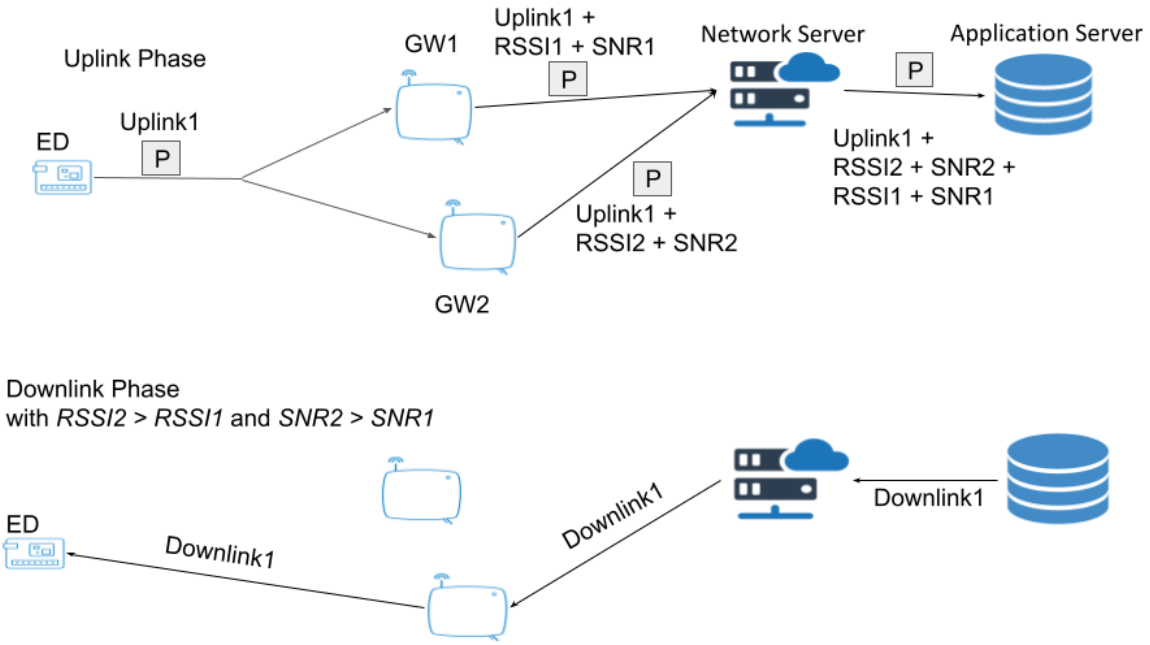


Figure 3.7: It illustrates two main features of LoRaWAN: deduplication and downlink gateway selection. When multiple gateways receive copies of the same uplink, the NS aggregates them and uses this information later to determine the best gateway to use to forward a downlink to an ED. In the top image, an uplink packet P is broadcasted by an ED and GW1 and GW2 receive it with respectively RSS1 and RSS2. In the bottom image, assuming RSSI2>RSSI1, the NS schedules a downlink message on GW2 since it had a better signal quality of the uplink P.

Options (FOpts) field of the FHDR, or, if the FOpts field is insufficient due to length constraints, within the frame payload using specifically Frame Port (FPort) 0.

MAC commands (Tab. 3.2) consist of a one-byte identifier, known as the Command Identifier (CID), followed by command-specific parameters. The ED and the NS use these commands to perform tasks such as adjusting the data rate, modifying transmission power, configuring channels, and managing device status. Examples of MAC commands include LinkCheckReq and LinkCheckAns for checking network connectivity, DutyCycleReq for setting duty cycle limitations, and RXParamSetupReq for adjusting reception parameters. The use of MAC commands allows the NS to optimize network performance by dynamically configuring EDs based on network conditions and policies. By sending MAC commands, the NS can instruct EDs to change their communication parameters, thereby improving network efficiency, reducing interference, and enhancing the reliability of data transmission within the LoRaWAN network.

3.2.4 Join Procedure

The join request/accept packets are part of the join procedure, started by an OTAA ED that wants to connect to the LoRaWAN network. During a join procedure, the ED will send an unencrypted packet with a payload containing DevEUI, JoinEUI, a value called DevNonce and a MIC. This DevNonce is an incremental value used by both parties to ensure that no attacker can use again the same packet, avoiding any kind of replay attacks. The MIC is instead used to check the integrity of the message, but we will cover later in Sec. 3.2.6 these and more security features used in LoRaWAN.

Table 3.2: List of MAC commands and their CIDs in LoRaWAN.

CID	MAC Command	Description
0x02	LinkCheckReq	End-device checks link quality with network server
0x02	LinkCheckAns	Network server responds with link status
0x03	LinkADRReq	Server requests to adjust data rate or power
0x03	LinkADRAns	End-device acknowledges ADR changes
0x04	DutyCycleReq	Server imposes duty cycle limitations
0x04	DutyCycleAns	End-device acknowledges duty cycle settings
0x05	RXParamSetupReq	Server sets RX window parameters
0x05	RXParamSetupAns	End-device acknowledges RX setup
0x06	DevStatusReq	Server requests device status
0x06	DevStatusAns	End-device reports battery and margin
0x07	NewChannelReq	Server adds or modifies channels
0x07	NewChannelAns	End-device acknowledges channel changes
0x08	RXTimingSetupReq	Server adjusts RX window timing
0x08	RXTimingSetupAns	End-device acknowledges timing setup
0x09	TxParamSetupReq	Server sets TX power parameters
0x09	TxParamSetupAns	End-device acknowledges TX settings
0x0A	DlChannelReq	Server specifies downlink channel
0x0A	DlChannelAns	End-device acknowledges downlink channel
0x0B	RekeyInd	End-device indicates protocol version change
0x0B	RekeyConf	Server confirms new protocol version
0x0C	ADRParamSetupReq	Server sets ADR algorithm parameters
0x0C	ADRParamSetupAns	End-device acknowledges ADR setup
0x0D	DeviceTimeReq	End-device requests current time
0x0D	DeviceTimeAns	Server provides network time
0x0E	ForceRejoinReq	Server forces device to rejoin network
0x0F	RejoinParamSetupReq	Server sets rejoin parameters
0x0F	RejoinParamSetupAns	End-device acknowledges rejoin setup

Table 3.3: List of possible MType values and their meaning. MType bits are part of the Mac Header of a LoRaWAN frame.

MType	Description
000	Join Request
001	Join Accept
010	Unconfirmed Data Up
011	Unconfirmed Data Down
100	Confirmed Data Up
101	Confirmed Data Down
110	1.0.x - Reserved for Future usage/1.1 - Uplink OTAA rejoin request
111	Proprietary - Custom defined operations implemented by the customer

Now the Join-request message has been received by the Network Server. If the ED is allowed to join the network, the NS will respond to the ED message with a Join-accept message. The Join-accept message is an encrypted packet that consists of the following fields:

- AppNonce – this is a random value or a unique ID provided by the network server. This value is used by the ED to derive the two session keys, AppSKey and NwkSKey.
- NetID – identifies the network ID.
- DevAddr – this is the 32-bit device address assigned by the network server. Consists of NwkID and NwkAddress and it is unique within the current network.
- DLSettings – this is a 1-byte sized field consisting of downlink settings which the end device should use.
- RxDelay – the delay between TX (transmission window) and RX (receiving window)
- CFList – contains the optional list of channel frequencies to be used for the end device

The ED will now decrypt the message, read the values of the payload, derives the session keys to communicate with the NS and it is now ready to exchange messages in the network.
Lastly, the rejoin request is used by an ED that is already part of the network to reset the session information and follows the same procedure of the join request, except that it is encrypted and contains only the NetID and the DevEUI, as the other information is not needed. It is then followed by a normal Join-accept message as we already saw before.

3.2.5 Adaptive Data Rate

LoRa network allows End Devices to use any of the possible data rates and transmission power, individually defined for each device. This feature is used by LoRaWAN to adapt and optimize the communication parameters in the network and is called Adaptive Data Rate (ADR). To enable it, the ED has to set a bit in the uplink frame header, called ADR bit, in this way the network will control the data rate and Tx power of the ED through the appropriate MAC commands. Otherwise, the NS leaves the ED free to use its own communication parameters, but it can try to activate the ADR itself by setting the ADR bit in the next downlink scheduled.

The ADR mechanism controls the following transmission parameters of an ED.

- Spreading factor
- Bandwidth
- Transmission power (Tx Power)

ADR helps optimize power consumption in IoT devices, which is crucial for their longevity while ensuring reliable message reception by the gateways. When ADR is in use, the NS will indicate to the end device that it should reduce transmission power or increase data rate depending on the quality of the communication. End Devices that are close to gateways should use a lower spreading factor and higher data rate, otherwise, they should use a high spreading factor as they need to cover longer distances, at the cost of slow communication and more power consumption.

To check that uplinks are still being received, the ED uses a counter called ADR_ACK_CNT. Whenever an uplink is performed, the ED increments ADR_ACK_CNT and, after reaching ADR_ACK_LIMIT (an implementation dependent variable) uplinks without any downlink from the NS, the ED explicitly requests an acknowledgement downlink by setting the ADR acknowledgement request bit. After another ADR_ACK_DELAY without any reply, the ED tries to regain connectivity by increasing the Tx power to an higher one, then switching to a lower data rate that provides a longer radio range. If again the ED does not receive a reply, it will lower its data rate again every time another ADR_ACK_DELAY uplinks are sent. If everything fails, it should reverse back to default channels, data rates and bandwidths.

3.2.6 Security features

The fundamental security properties supported by LoRaWAN are mutual authentication, integrity protection, and confidentiality.

Mutual Authentication

Mutual authentication is straightforward because the NS must know in advance the hardcoded information of the device, root keys, JoinEUI and DevEUI for OTAA devices and session context for ABP devices. It is crucial since this information is included in messages from the beginning of the communications. Even in the initial join message sent by an OTAA device, although the entire packet is sent in plaintext, the MIC appended at the end is cryptographically derived using root keys already known by the server. This allows the server to ensure the authenticity and integrity of the join message simultaneously. The subsequent join-accept packet is entirely encrypted with the same shared root key, assuring the device that the response originates from the genuine NS. The allocation of EUI-64 identifiers, used for JoinEUI and DevEUI, requires these unique identifiers to be issued by a central authority, in this case, the IEEE Registration Authority. On the other hand, LoRaWAN networks are identified by a 24-bit globally unique identifier assigned by the LoRa Alliance.

Integrity

To ensure the integrity of communication, a cryptographic MIC is calculated over the MHDR and payload of the message. For regular uplink and downlink messages, the MIC is generated using a specific key and the AES-CMAC algorithm, which is a block cipher-based message authentication code. The output of AES-CMAC is then truncated to the first 4 bytes and appended to the end of the message. When the NS receives the message, it first verifies the MIC. If the verification is successful, the NS decrypts the packet, though the payload remains encrypted with the AppSKey which the NS does not have access to, and then forwards the message to the application server. During the Join procedure, since the appropriate session keys have not been generated yet, the MIC is calculated using the root AppKey instead.

Confidentiality

LoRaWAN implements end-to-end encryption for application payloads exchanged between EDs and ASs [22]. This approach is founded on the understanding that, in most scenarios, LoRaWAN network

providers are separate entities from the organizations that own the ASs. Much like mobile network operators that function merely as conduits for data without accessing its content, LoRaWAN ensures that network operators can not read the transmitted messages. While upper-layer protocols such as TLS have been developed to secure confidentiality over untrusted networks, these solutions are not ideally suited for the IoT domain. In IoT environments, the addition of security layers can lead to increased power consumption and complexity—critical constraints for resource-limited devices. Therefore, LoRaWAN relies on standardized AES cryptographic algorithms for its security mechanisms. Specifically, it employs the AES cipher combined with multiple modes of operation: Cypher-based Message Authentication Code (CMAC) for integrity protection and Counter-mode encryption (CTR) for effective message encryption. Consequently, all LoRaWAN traffic is secured using different session keys, depending on the message type, one key for the payload and others for securing the entire packet content after payload encryption. Each payload is encrypted using AES-CTR and includes a MIC calculated with AES-CMAC to prevent packet tampering.

In LoRaWAN version 1.0, only two distinct session keys were utilized: AppSKey and NwkSKey, responsible for encrypting the payload and the entire packet, respectively. With the advent of LoRaWAN 1.1, the key management mechanism was significantly enhanced, defining several different keys, each specific to a type of communication as shown in Fig. 3.4. These keys, NwkSEncKey, SNwkSIntKey, FNwkSIntKey, and AppSKey, are AES-128 keys, providing sufficient cryptographic strength to resist common attacks on encrypted payloads. Moreover, these keys can be refreshed in a pseudorandom manner whenever an ED initiates a join or re-join request, thereby adding an extra layer of security through key renewal.

Counters

To safeguard against replay attacks, each message incorporates an incremental counter, with separate counters allocated for different types of messages (e.g., one for uplinks, another for downlinks, and a third for join requests). These counters enable the system to track received messages and discard any potential replay attempts by malicious actors or unintended retransmissions. After verifying the MIC and decrypting the message, the receiver examines the counter value. It expects this value to be *at least* equal to the last counter encountered plus one, accepting any message with a counter strictly greater than the previous one received. The implementation of incremental counters is a novel feature introduced in LoRaWAN 1.1. In earlier versions, these counters were random values that both the ED and the NS were expected to remember. However, this method proved impractical, as it is unfeasible to track thousands or millions of different nonces, which could eventually be replaced by newer values by the receiver, thereby enabling replay attacks.

3.2.7 Security Weaknesses

Protocol Weaknesses

LoRaWAN's security features have been problematic since the initial release of the standard. Over the years, numerous security issues have emerged. The vulnerabilities in LoRaWAN security predominantly affect the communication between end devices and the network server [23, 24], as the ED is often considered the weakest link in the communication chain, potentially leading to indefinite disconnection from the network. The inherent security weaknesses of ABP devices, stemming from

the static nature of their session keys, can be exploited in several ways [25].

These challenges prompted the release of a significant update to the standard, LoRaWAN 1.1, which introduced substantial security enhancements to address the deficiencies of the 1.0.X version. Nevertheless, LoRaWAN 1.1 remains imperfect, as not all issues have been resolved [26], and new vulnerabilities have been identified in recent years [27]. Moreover, it allows the NS to downgrade and operate using a 1.0.X version, thereby reintroducing vulnerabilities from the previous standard [26]. LoRaWAN security concerns are not confined solely to the protocol itself but also to the behaviour of devices within the network, which can inadvertently leak confidential information [28], or the network structure, with its centralized nature and trust-based operations posing multiple threats for the well-being of the network.

Centralization

The architecture of LoRaWAN follows a star-of-stars topology (Fig. 3.2), where EDs communicate with the NS through GWs. In this setup, the NS acts as the central authority managing data routing, device authentication, and security functions such as MAC command processing and key management. The GWs function as passive relays, simply forwarding messages between the EDs and the NS. This centralized design creates a highly trust-dependent model. The NS is assumed to be reliable and secure, as it manages sensitive operations, including the verification of MIC and the distribution of encryption keys. GWs instead are often deployed in unprotected environments, and the assumption is that they do not tamper with the data they forward. This trust in the backbone components can be easily exploited by attackers, posing substantial security risks.

One of the primary weaknesses of LoRaWAN's centralized architecture is the NS, which serves as a single point of failure. The NS handles all critical network functions, including device management, data routing, and security operations. If the NS is compromised or disrupted, it can render the entire network inoperative, exposing several key vulnerabilities:

- **Network Downtime**: A compromised or failed NS can cause all connected devices to lose communication, leading to a complete halt in IoT operations.
- **Security Breaches**: If an attacker gains control of the NS, they can manipulate network traffic, intercept sensitive data, and exploit session keys to impersonate devices.
- **Scalability Issues**: As the network grows, the NS can become a bottleneck, struggling to handle increased traffic and device management tasks effectively.
- **Reliability Concerns**: Any attack or malfunction targeting the NS compromises the overall stability of the network, leading to service interruptions and inconsistent performance.

Chapter 4

Bluetooth Low Energy Nodes Detect, Enquire and Recognition

4.1 Introduction

Research in network security and privacy continues to be a highly active area, particularly in the context of the Internet of Things (IoT) [29–31]. Bluetooth Low Energy (Bluetooth Low Energy (BLE)) plays a critical role in this domain, owing to its widespread implementation in everyday devices like smartphones, fitness trackers, and wearables. Moreover, BLE facilitates the creation of both ad-hoc mesh networks, as well as more formalized, standard-defined mesh networks [16], offering the capability to exchange data across different platforms efficiently.

Since version 4.0 of the BLE standard [4], privacy and security concerns related to Medium Access Control (MAC) address handling have undergone significant revisions by device manufacturers to mitigate tracking and profiling risks for users [32]. A key improvement introduced by the standard is the randomization of MAC addresses, which are rotated approximately every 15 minutes to protect users from tracking. However, this safeguard is not universally implemented correctly. Furthermore, even when it is correctly executed, certain devices expose data that can be captured with low-cost hardware, enabling potential fingerprinting and tracking of these devices [33].

Despite the rotating MAC addresses, attackers can leverage this exposed data to generate a unique fingerprint, a sort of identifier, for a device, allowing them to track the same physical device across different times and locations, despite the dynamic MAC addresses. This concept of fingerprinting is well established in network security, having been used extensively in web technologies for tracking anonymous users. For instance, websites gather various user settings and characteristics, such as screen resolution, language, and hardware information, to build a unique fingerprint for each user, allowing tracking even without knowledge of their true identity. The user, although anonymous, becomes traceable over time using this fingerprint.

Similarly, in the BLE context, even if a device's MAC address is randomized, it may still reveal hardware-specific details, such as clock skews [34], software versions, or vendor information. These details can be broadcasted or made available for retrieval by any nearby device, without requiring any special permissions or authentication from the target device. A scanner can thus extract sufficient data from the target, in compliance with the BLE standard, to create a unique fingerprint, all without directly breaching security protocols.

In this work, presented in [35] and extended in [36], we present BLENDER, a proof-of-concept system capable of analyzing, fingerprinting, and tracking BLE devices. By employing BLENDER, we demonstrate the feasibility of enumerating and fingerprinting BLE devices, and we also showcase the potential for large-scale monitoring and tracking systems using off-the-shelf hardware and specialized software. BLENDER exploits key components of the BLE standard, such as Generic Access Profile (GAP) and Generic Attribute Profile (GATT), which are integral to detecting, connecting, and exchanging information between devices. Through these mechanisms, we retrieve relevant data for analysis.

Our proposed system enables the discovery, enquiry, and fingerprinting of active BLE devices in real-time. We will outline the techniques and solutions adopted in BLENDER, as well as suggest potential enhancements for future iterations. Additionally, we will present the results from three experimental deployments of BLENDER in real-world environments, demonstrating its original capability to detect and fingerprint BLE devices across various locations.

The remainder of this chapter is organized in the following manner:

In Section 4.2, we provide an overview of key concepts related to BLE device fingerprinting, crowd counting, and associated privacy risks. This sets the stage by explaining how BLE advertising and scanning processes can be exploited to identify and track devices despite privacy mechanisms like MAC address randomization.

Section 4.3 introduces the BLENDER system, explaining its architecture and four detection strategies: *PassiveListening*, *ScanTrigger*, *Enquiring*, and *JustStore*. Each strategy leverages specific BLE characteristics to collect data, allowing the system to fingerprint devices even when MAC address randomization is used.

In Section 4.4, we demonstrate the real-world effectiveness of BLENDER by discussing results from field experiments. The system's ability to track devices under various conditions is highlighted, showcasing its potential for monitoring in both indoor and outdoor environments.

Finally, in Section 4.5, we summarize key findings and emphasize the limitations of MAC address randomization as an effective privacy measure in BLE devices. The findings underscore the need for future research and development efforts to provide more robust security solutions as IoT technologies continue to expand.

In summary, the BLENDER system provides a detailed insight into the vulnerabilities of BLE devices, showcasing how a relatively simple setup can be used to track and monitor devices. This project underscores the importance of strengthening security mechanisms in BLE to protect against the risks posed by enumeration and fingerprinting attacks, especially as the number of IoT devices continues to grow. The lessons learned from this research will serve as a foundation for future improvements in Bluetooth security, particularly in enhancing privacy protections for end users.

4.2 Background

BLE (versions 4.2, 5.0, and 5.1) operates in the 2.4 GHz Industrial, Scientific, and Medical (ISM) band, ranging from 2.400 to 2.4835 GHz. There are 40 allocated RF channels, each with a separation of 2 MHz. In BLE, devices may act as advertisers, broadcasting advertising packets using three designated advertising channels (37, 38, and 39). These channels are used to reduce interference by spreading the broadcast across multiple frequencies. The advertising intervals, known as Advertising

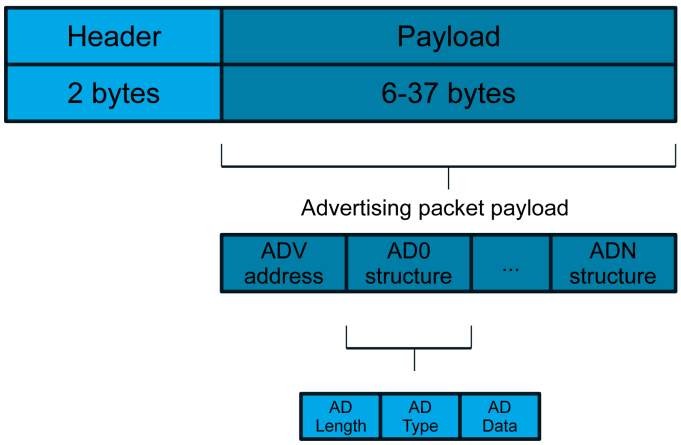


Figure 4.1: Advertising channel PDU format of ADV_IND BLE advertising packet [37].

Intervals, allow devices to broadcast packets systematically, which can be detected by other BLE-enabled devices via their firmware and software stacks. The remaining 37 channels are reserved for data transmission, used in paired communications between devices once a connection has been established. Upon connection, devices transition to the role of Broadcasters, advertising their presence and services.

There are various types of advertising Protocol Data Unit (PDU) packets used for different communication purposes:

- Advertising PDUs: (ADV_IND, ADV_DIRECT_IND, $ADV\_NONCONN\_IND$, ADV_SCAN_IND) are used to signal the services or data offered by the sender.
- Scanning PDUs: ($SCAN\_REQ$, $SCAN\_RSP$) are used to request permission to scan another device for specific services.
- Initiating PDUs: ($CONNECT\_REQ$) are used to initiate connections between devices.

The structure of an advertising packet (e.g., ADV_IND) is depicted in Figure 4.1.

4.2.1 GAP

GAP is a fundamental protocol that defines how BLE devices interact and establish connectivity within a network, structured into four distinct roles. In connectionless communication, the Broadcaster role allows devices to continuously emit advertising packets (ADV packets) containing key information such as the device's identity, services, and metadata. These packets are transmitted through designated advertising channels, crucial for discovering devices in scenarios like indoor localization, where BLE Beacons broadcast their presence to nearby Observers. Observers are passive devices that listen to advertising channels for ADV packets, gathering information to detect nearby Broadcasters and assess their advertised features and services. This one-way communication system enables devices like beacons to broadcast without expecting a response, ideal for tracking and proximity-based applications.

In connection-oriented mode, devices adopt the roles of Central and Peripheral. Typically, a Peripheral device has limited power and processing resources, such as wearables, sensors, or smart health devices like heart rate monitors and fitness bands. A Peripheral periodically advertises

its availability using ADV packets. Once detected by a Central device, usually more resourceful like a smartphone or computer, the Central can initiate a connection. GAP defines the process by which Centrals and Peripherals establish these connections. After the Central detects a Peripheral's advertisement, it can request to connect. If accepted, the devices transition into an interactive communication phase, exchanging data using the connection-oriented mode. The Central can retrieve data from the Peripheral and send commands or configuration instructions, commonly used in applications like fitness tracking where a smartphone retrieves real-time data from a connected wearable.

GAP includes mechanisms to control and optimize communication, such as the advertising interval, which dictates how frequently a Broadcaster transmits ADV packets. A shorter interval increases the chances of device discovery but consumes more power, making it essential to balance advertising frequency with energy efficiency for low-energy devices to ensure long battery life. Additionally, the scan request/scan response mechanism allows an Observer to actively request more information from a Broadcaster. When an Observer detects a scannable Broadcaster, it can send a scan request packet, prompting the Broadcaster to respond with a scan response packet that provides detailed information about the device, its services, and status, targeted to the specific Observer. A critical aspect of GAP is the seamless transition from connectionless to connection-oriented roles. When a Broadcaster is connectable and an Observer can initiate connections, they can switch roles: the Observer becomes the Central, and the Broadcaster becomes the Peripheral. This switch enables the establishment of a bidirectional communication channel, facilitating more complex data exchange.

4.2.2 GATT

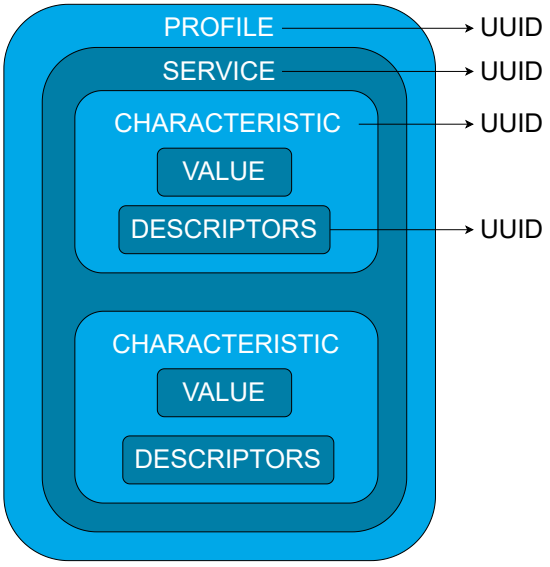


Figure 4.2: The nested object structure used in GATT.

GATT organizes communication into logical entities known as Services and Characteristics. Each GATT interaction is structured by the Attribute Profile (ATT) protocol, which assigns Services and Characteristics unique 16-bit Universally Unique Identifier (UUID) identifiers. These identifiers ensure consistent access to specific data across devices. Unlike the connectionless mode,

GATT enables full-duplex, bidirectional communication, allowing both devices to exchange data simultaneously for richer interactions. Within the GATT framework, devices assume the roles of Server or Client. The Server hosts data through Profiles, Services, and Characteristics, as shown in Fig. 4.2. Profiles, defined by the Bluetooth Special Interest Group (SIG) or manufacturers, are composed of group related services. For example, the standardized Heart Rate Profile used in fitness devices includes the Heart Rate Service for monitoring heart activity and the Device Information Service for details like manufacturer and model. Services are composed of Characteristics, which represent specific data pieces. Characteristics can range from a few bytes to larger values and may be read-only or read-write. For instance, a sensor characteristic might be read-only to display data, while another could allow user adjustments like sensor sensitivity. Attributes are the lowest level of the GATT structure, representing the actual data exchanged between devices. Characteristics abstract over Attributes, presenting data meaningfully as sensor readings, configuration information, or user properties. GATT operates over dedicated connections, ensuring private and secure data exchange, unlike the broadcast nature of connectionless communication. The ability to define custom Profiles and Services adds significant flexibility to the BLE ecosystem. By organizing data through Services and Characteristics, GATT maintains high interoperability among BLE devices while supporting specialized and customizable use cases.

4.2.3 MAC address randomization

The MAC address in BLE, composed of 6 bytes, is designed to rotate periodically to protect user privacy, with an indicative rotation period of 15 minutes. However, the actual implementation of this randomization can vary across devices. Older or non-compliant devices may either not implement randomization or generate static addresses that persist across sessions. Public MAC addresses are permanent identifiers assigned by the SIG Alliance, where the first three bytes represent the manufacturer, while private addresses can either be resolvable or non-resolvable. A resolvable random address allows bonded devices to recognize each other, even after MAC address rotation, but prevents tracking by unbonded devices.

Smartphones generally adhere to these standards, systematically randomizing MAC addresses. However, many other BLE devices, such as smart bands and headphones, may utilize static addresses to optimize power consumption and ease reconnections, making them particularly susceptible to tracking. In the BLENDER system, both static and randomized MAC addresses are considered for crowd monitoring. Although public or random address types are detected and temporarily stored, device counting by design is not influenced. It is also important to note that devices implementing MAC address randomization do not rotate too frequently, as this would introduce overhead. Thus, the most recent devices generally comply with the recommended 15-minute interval, making MAC-based tracking more challenging. When MAC addresses are randomized, fingerprinting techniques can still be used to uniquely identify devices by analyzing other consistent data.

4.2.4 Privacy concerns

In BLENDER, privacy is maintained by only transmitting pseudonyms or fingerprints related to detected devices rather than MAC addresses, which fall under privacy regulations like the European Union's GDPR. However, a malicious actor could develop a system similar to BLENDER that

collects actual MAC addresses or user data scannable via the ATT protocol, allowing for detailed tracking. This presents a significant privacy concern, as such a system could be used by national entities to track a large portion of the population, particularly given the widespread adoption of BLE technology. The inherent risks associated with the pervasive use of BLE in everyday devices make it essential to address these privacy vulnerabilities proactively.

4.2.5 BLE Fingerprinting techniques

Due to its widespread adoption, BLE has become a focal point for research on both offensive and defensive security techniques. Numerous studies have analyzed the privacy mechanisms of BLE, particularly MAC address randomization, revealing its shortcomings as a method to ensure device anonymity. While intended to protect against unauthorized tracking, research has consistently demonstrated how attackers can bypass this mechanism, exploiting static elements of the protocol to identify and track devices, raising serious concerns about the effectiveness of BLE's privacy features.

Celosia et al. [38] introduced a method for tracking BLE devices by leveraging their GATT profiles, bypassing the privacy mechanism of MAC address randomization. While MAC addresses are randomized to prevent tracking, the static nature of GATT profiles, which contain device-specific services and characteristics, enables the identification and tracking of individual devices. The authors demonstrate how the anonymity set can be reduced to uniquely identify devices, revealing significant privacy risks despite MAC randomization.

Issoufaly et al. [39] explored the vulnerabilities of BLE devices, highlighting that many fail to implement MAC address randomization correctly, leaving them exposed to tracking attacks. They introduced BLEB, a botnet of compromised BLE devices, which can track users by passively collecting data from advertising packets. Even when randomization is used, information such as static addresses or UUIDs can still leak, enabling tracking.

Das et al. [40] focused on fitness trackers and found that many devices broadcast static BLE addresses, making them vulnerable to tracking. The paper revealed that top brands, such as Fitbit and Jawbone, often fail to implement MAC randomization, exposing users to potential privacy breaches by allowing attackers to continuously monitor BLE traffic.

Becker et al. [41] proposed an address-carryover algorithm that exploits the asynchronous behavior of MAC address changes and static identifiers in BLE advertisements. This method allows tracking across randomization cycles by correlating unchanged payload elements with newly randomized addresses. The study revealed that popular operating systems, including iOS, macOS, and Windows 10, are vulnerable to this attack, as they broadcast static identifiers that can be used to link randomized addresses back to the original device.

In their later work, Celosia et al. [42] delved deeper into the limitations of MAC address randomization in BLE advertising. They identified weaknesses in how randomization is implemented, allowing attackers to exploit static identifiers in the advertising payload to re-identify devices across multiple sessions. This persistence of static tokens in the payload, despite address randomization, continues to pose significant privacy risks for users.

Together, these works illustrate the persistent vulnerabilities in BLE's privacy mechanisms, particularly around MAC address randomization. While designed to enhance privacy, the static nature of certain elements within the BLE ecosystem, like GATT profiles, UUIDs, and other identifiers,

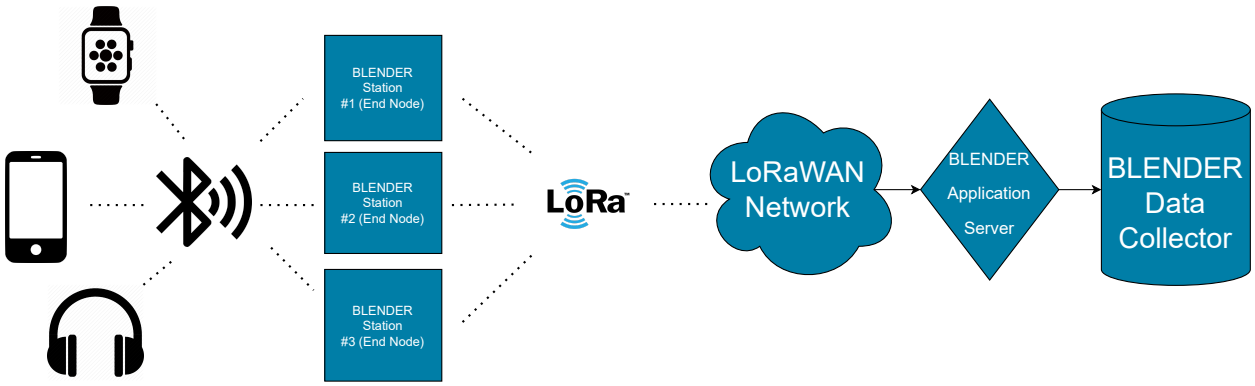


Figure 4.3: The architecture of the BLENDER system consists of independent station nodes (stations) that collect data autonomously and transmit it to a centralized cloud server via a LoRaWAN network. This ensures long-range, low-power communication, minimizing battery consumption.

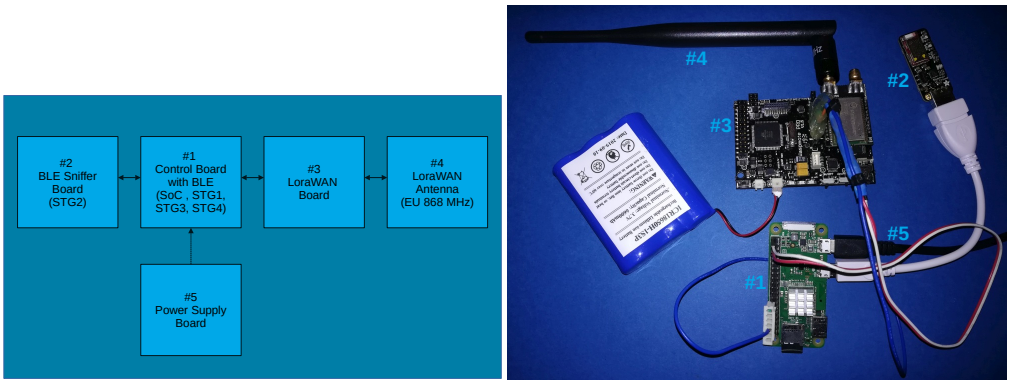
leaves devices vulnerable to tracking, necessitating more robust solutions to protect user privacy.

4.3 BLENDER

BLENDER is an automatic system composed of autonomous station nodes (referred to as "stations"), relying solely on their own resources for computing, power, and networking over both short and long ranges. Stations can be deployed at various locations, distanced from each other, such as at the entrances of buildings or in public areas. Indoors, stations can be separated by 30 to 50 meters, while outdoors, they can cover distances of up to 100 meters. The stations are capable of autonomously collecting data for crowd monitoring and tracking, employing four distinct strategies. Specifically, each BLENDER station implements two different strategies to discover the MAC addresses of BLE devices. Once a device's MAC address is detected, the station calculates the fingerprint of the device, when possible. This is particularly useful for tracking purposes, as it addresses the challenge posed by BLE MAC address randomization by establishing connections to read device attributes for fingerprinting. Additionally, only for devices that use static MAC addresses, a connection-less fingerprinting of the MAC address itself is used. These processes are continually repeated by each station, which generates a LoRaWAN frame every minute. This frame contains data regarding the number of discovered devices and any fingerprints that have been computed.

4.3.1 System Architecture

Fig. 4.3 illustrates the overall system architecture, showing how station nodes transmit aggregated data via long-range uplink connections. These uplinks are essential for retrieving real-time data from BLENDER stations, which may be spread across extensive geographic areas, such as an entire smart city. Various options for long-range communication were assessed, both in open-source and commercial contexts. Given that station nodes can be situated 2 to 10 kilometers from collectors or gateways, technologies like 5G/4G LTE, NB-IoT, LoRaWAN, and SigFox were evaluated as potential solutions, balancing cost, efficiency, and coverage. The option of using WiFi for communication in urban areas was considered, but it presented challenges related to coverage in areas without open networks and high power consumption, making it unsuitable for IoT deployments. While cellular networks offered affordable and compact hardware solutions via SIM cards, the costs of



(a) BLENDER station diagram, showing the different components needed from the BLE sniffer to the LoRaWAN board

(b) BLENDER station setup and its components. Even in its PoC state it shows how little a BLENDER station can be.

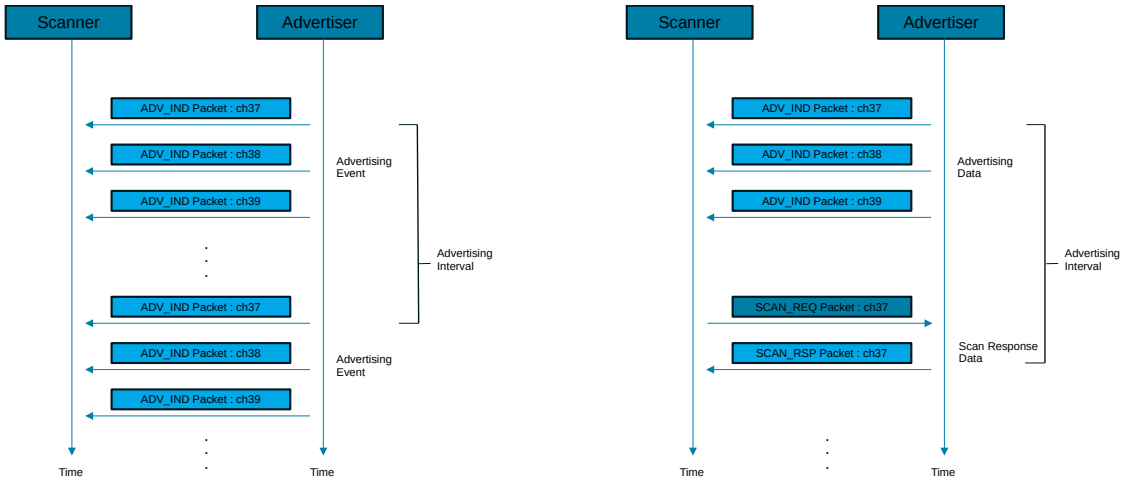
Figure 4.4: BLENDER system.

acquiring and managing these cards, along with coverage issues in non-urban areas, rendered this approach impractical. Ultimately, Low-Power Wide-Area Network (LPWAN) technologies like NB-IoT, SigFox, and LoRaWAN were considered, with LoRaWAN being selected for its low cost and ability to efficiently cover the required distances, making it the best choice to transmit aggregated data to the cloud for collection and storage.

The BLENDER station's architecture is illustrated in Fig. 4.4a, where each hardware and software block is detailed. At the core is the Broadcom BCM2835 System-on-Chip (SoC) housed in a Raspberry Pi Zero W, which runs a lightweight Debian-based Linux system. The station uses Java, Python, and shell scripts to implement the different stages of BLENDER. For a specific tecnique, called $ScanTrigger$, additional hardware (Adafruit Bluefruit LE sniffer) is required to capture BLE traffic. Two long-range uplink options were evaluated: Libelium Waspmote PRO 1.5 and Heltec Cubecell AB01, both paired with a LoRaWAN antenna. Power is supplied by a Waveshare UPS HAT, equipped with a 3.7V 1000mAh LiPo battery and solar panel support. Fig. 4.4b displays a station setup used in the laboratory with Waspmote PRO as the LoRaWAN board.

The BLENDER system operates as a distributed application on the The Things Network (TTN) platform, where each station is registered as an End Devices (ED), enabling stations to transmit uplink data to the cloud, where it can be accessed by the BLENDER data collector. TTN is a global LoRaWAN network tailored for IoT devices, allowing users to create custom applications while utilizing the public infrastructure. Public LoRaWAN gateways act as the network's endpoints, receiving uplinks from end-devices, like the BLENDER stations.

For data collection, the BLENDER Data Collector node runs on a Raspberry Pi 3B+ with internet connectivity. The application communicates with TTN via MQTT, subscribing to notifications about uplink payloads received from BLENDER stations. The data is then stored in a relational database, enabling further analysis and monitoring of collected data. This distributed setup facilitates real-time data aggregation and processing across widely dispersed stations, ensuring efficient, scalable, and low-power data collection.



(a) Passive scanning mechanism. **(b)** Active scanning mechanism.

Figure 4.5: Different scanning mechanisms in BLE.

4.3.2 Detect and count devices

Algorithm 1 Passive listening, used to discover and count BLE devices in the area.

1: Define $\mathcal{T}$ as the time length of a BLE advertisement scan
2: Define $\mathcal{B}$ as a storage buffer for the MAC addresses found
3: **while** $\mathcal{T}$ is not over
4: Sense the advertising channels to detect ADV packets
5: Detect the MAC addresses of BLE devices in range $\mathcal{P}$
6: Store $\mathcal{P}$ in $\mathcal{B}$
7: **end while**

Passive Listening

In this approach, described in Algorithm 1, the scanner device continuously listens to advertising channels to detect incoming packets from Broadcaster devices. These devices may be peripherals, beacons, or even smartphones broadcasting advertising packets, such as those used for contact tracing. Advertising packets contain the advertiser's MAC address. The passive listening strategy involves gathering these addresses during the scan interval and collecting all unique addresses detected. This method adheres to the BLE passive scanning model, as depicted in Figure 4.5a.

Scan Trigger

The BLE advertising mechanisms, illustrated in Figure 4.5b, can be leveraged using an innovative method we call the $SCAN\_REQ - SCAN\_RSP$ packet exchange. Advertising PDUs hold up to 31 bytes of data, typically broadcasting the advertiser's presence and available services. In some use cases, the Observer device, upon receiving an ADV packet, may want more detailed information from the Broadcaster. The BLE standard supports this by allowing the Observer to send a $SCAN\_REQ$ packet to the Broadcaster's MAC address. The Broadcaster responds with a $SCAN\_RSP$ packet, which can contain up to 31 additional bytes of data.

The format of the scan request PDU is shown in Figure 4.6. The $ScanTrigger$ strategy is detailed in Algorithms 2 and 3. The first method, Embedded Active Scanning (EAS), relies on identifying

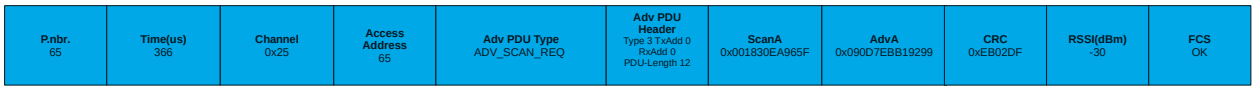


Figure 4.6: *SCAN_REQ* packet format. *ScanA* is the scanning device MAC address.

scanner MAC addresses in *SCAN_REQ* packets by using a scannable BLE device under our control, part of the station node. The second, Opportunistic Active Scanning (OAS), capitalizes on scannable Broadcasters in range and captures *SCAN_REQs* sent by other devices interacting with these Broadcasters. Both strategies can be used individually or together for enhanced efficiency.

In EAS, BLENDER acts as a decoy to identify and collect the MAC addresses of scanning devices sending *SCAN_REQ* packets. However, the scannable device can potentially be recognized by targets unless it employs randomized address rotation. On the other hand, OAS is entirely passive and stealthy, as it captures the *SCAN_REQ* traffic between external devices without requiring a scannable onboard Broadcaster, making it undetectable by target devices.

Algorithm 2 *ScanTrigger* - EAS

1: Define $\mathcal{T}$ as the time length of a BLE advertisement scan
2: Define $\mathcal{B}$ as a storage buffer for the MAC addresses found
3: **while** $\mathcal{T}$ is not over
4: Broadcast advertising packets which contain references to additional data stored in the embedded device
5: Wait for possible incoming *SCAN_REQ* packets to embedded scannable device
6: Detect the MAC addresses of BLE scanning devices in range $\mathcal{P}$
7: Store $\mathcal{P}$ in $\mathcal{B}$
8: **end while**

The *PassiveListening* and *ScanTrigger* strategies can be used to monitor BLE devices by leveraging passive and active scanning methods defined in the BLE standard. The innovative aspect of *ScanTrigger* lies in its ability to detect devices that, while not advertising, are performing active scans due to operating system or firmware behavior. These techniques provide sufficient detection and counting capabilities for crowd-monitoring purposes.

In practice, the most monitored advertising packet is *ADV_IND*, commonly broadcast by peripherals, while *SCAN_REQ* is the key packet used for data collection in *ScanTrigger*. Table 4.1 provides details on the various advertising and scan request packets, explaining their signatures and the applicability of the strategies for detecting devices transmitting these packet types.

PDU Type	Packet name	Description	Passive Listening	Scan Trigger	Enquire	JustStore
0000	*ADV_IND*	Connectable Undirected Advertising	✓		✓	✓
0001	*ADV_DIRECT_IND*	Connectable Directed Advertising	✓			✓
0010	*ADV_NONCONN_IND*	Non-Connectable Undirected Advertising	✓			✓
0011	*SCAN_REQ*	Scan Request		✓	✓	✓
0100	*SCAN_RSP*	Scan Response	✓			✓
0101	*CONNECT_REQ*	Connection Request				
0110	*ADV_SCAN_IND*	Scannable Undirected Advertising	✓			✓
0111-1111	Reserved					

Table 4.1: PDU type codes from header described in BLE specification v4.2 [43] and their description.

4.3.3 Enquiring and Fingerprinting Strategies

Fingerprinting a device becomes feasible only when sufficient unique data is captured from the target. Enquiring expands on this by exploiting the BLE standard, this time using GATT instead of GAP, which was previously used for device discovery and counting in the strategies seen before.

Algorithm 3 *ScanTrigger* - OAS

1: Define $\mathcal{T}$ as the time length of a BLE advertisement scan
2: Define $\mathcal{B}$ as a storage buffer for the MAC addresses found
3: **while** $\mathcal{T}$ is not over
4: Scan advertising channels for $SCAN\_REQ$ packets addressing external devices nearby
5: Detect the MAC addresses of BLE scanning devices in range $\mathcal{P}$
6: Store $\mathcal{P}$ in $\mathcal{B}$
7: **end while**

While GAP involves passive or active scanning and operates via broadcast packets without requiring a connection, GATT comes into play once a connection is established between devices.

GAP works by broadcasting advertisements, with any nearby device capable of passively listening to this communication, regardless of filters applied to scan requests or responses. In contrast, GATT is used when a device transitions from broadcasting to an established connection, allowing more significant data exchange. Once a connection is formed, the central and peripheral devices can exchange information structured around Profiles, Services, and Characteristics. These are hierarchical containers of data, which together form a nested object structure as represented in Figure 4.2.

The detailed data transmitted through GATT can be used for more precise fingerprinting, as it encapsulates specific device properties and characteristics, enabling a more targeted and detailed analysis of the device's unique features.

There are several predefined services within BLE that are useful for fingerprinting based on static data. One particularly valuable service is the Device Information Service, which provides information such as the device vendor, model, firmware version, and serial number—data that typically remains unchanged and is therefore useful for identification. The TX Power Service is another valuable tool, as it reveals the transmission power settings, which are mainly static.

Enquiring involves making connection attempts and retrieving this type of static data through GATT operations. Fingerprinting combines the retrieved data from advertising, profiles, services, and characteristics, helping to uniquely identify a specific device by reducing the likelihood of collision between similar devices, even in environments with MAC address randomization.

Enquiring

In this phase, described in Alg. 4, the goal of the BLENDER station is to establish a connection with the target device and retrieve data through GATT interactions that can be used to create a device fingerprint. This active approach to fingerprinting focuses on gathering static, informative data rather than relying on hardware parameters related to transmission, such as advertising timing, time skews, or pseudo-random variations in MAC address rotation.

The essence of the *Enquiring* phase involves collecting structured information that is encapsulated in profiles, services, and characteristics available for reading once a connection is established. After successfully connecting to the target device, the station's scanner identifies the available characteristics and gathers relevant data, including the device's generic name, manufacturer, model, revision details, and other information that may vary depending on the device's configuration and the manufacturer's specifications.

The process of scanning device characteristics is facilitated by an automatic analysis tool based

Algorithm 4 Enquire

```
Define 𝒟 as the list of connectable devices found during detection phase
Define ℱ as the list of calculated fingerprints
Define ℬ as a temporary buffer
Define ℋ as an hash function
for D in 𝒟
    if Connection with D is successful
        for 𝒮 in D.services()
            for 𝒞 in 𝒮.characteristics()
                if Read of 𝒞 is successful
                    ℬ.append(𝒞.getContent())
                end if
            end for
        end for
        ℱ.append(ℋ(ℬ))
    end if
end for
```

on the *BLEAH* software, which is part of the Bettercap suite[1]. This tool is controlled by the main application running on the BLENDER station. It performs the task of retrieving relevant parameters, identifying the presence of characteristics and their values, or attributes that are crucial for creating a unique device fingerprint. The goal is to generate a fingerprint with a low probability of collision, even between devices of the same manufacturer and model.

The essential step in this process involves enumerating the device's attributes and handles, extracting relevant information, and combining them into a unique string. This aggregated data is then hashed using an algorithm designed to ensure a low likelihood of collisions. In BLENDER, we use the SHA-1 algorithm to create the fingerprint, which generates a 20-byte hash. However, due to the limitations imposed by LoRaWAN on the size of payloads in uplink frames, we employ only the first 5 bytes of the 20-byte SHA-1 hash as the device's fingerprint. This creates a short fingerprint that is compact enough for long-range transmission. Given this approach, it is essential to evaluate the probability of collision between 5-byte fingerprints derived from the 20-byte SHA-1 hash to ensure that each fingerprint remains sufficiently unique across different devices. Recalling the birthday paradox[2], the probability of a hash collision can be estimated. For a 5-byte (40-bit) hash, the probability of a 50% collision is reached after $2^{40/2} = 1,048,576$ hashes. This means the likelihood of a collision happening is less than $1 * 10^{-6}$. In simpler terms, it would take over one million different hashes for there to be a 50% chance of a collision, which is an extremely low risk. Moreover, this already small risk can be reduced even further by using techniques like data cleaning to remove potential collisions. Therefore, for practical use, the impact of collisions is negligible.

JustStore

The fourth strategy, referred to as JustStore, described in Algorithm 5, is used to fingerprint devices that expose public or static MACs. These addresses are well-suited for fingerprinting directly based on GAP-recovered data without requiring any active connection. JustStore is a connectionless, static address-driven, software-based BLE fingerprinting technique. In this method, the discovered static MAC address is directly fed into the SHA-1 hash function, and a subset of 5 bytes from the resulting hash value is used as the device's fingerprint. Since static addresses do not change over time and are

[1] https://github.com/bettercap/bettercap
[2] https://doi.org/10.1111/j.1740-9713.2007.00246.x

guaranteed to be unique, they serve as ideal input data for creating reliable fingerprints. Although static addresses are typically used by specific devices, such as smart bands/watches, wearable smart sensors or headphones, these devices are popular and widespread, making this residual case of static address presence an effective and valuable strategy for device identification and tracking.

Algorithm 5 JustStore

1: Define $\mathcal{D}$ as the list devices found during detection phase having public/static MAC address
2: Define $\mathcal{MAC}(\mathcal{D})$ as the function that returns the MAC address of device D
3: Define $\mathcal{F}$ as the list of calculated fingerprints
4: Define $\mathcal{B}$ as a temporary buffer
5: Define $\mathcal{H}$ as an hash function
6: **for** D in $\mathcal{D}$
7: $\quad\mathcal{F}$.append($\mathcal{H}(\mathcal{MAC}(\mathcal{D}))$)
8: **end for**

4.4 Data Collection, Experiments and Analysis

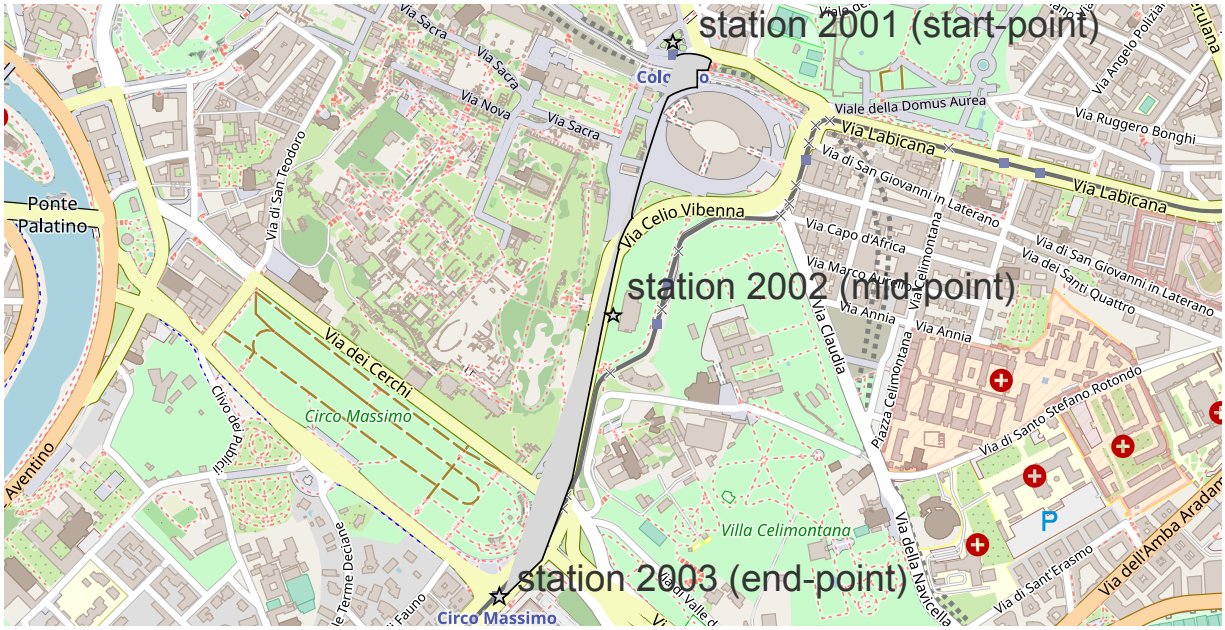


Figure 4.7: Path inference using fingerprint collected data in the city of Rome.

The payloads collected by each station node are transmitted every minute. This data includes both the count of BLE-enabled devices discovered in the previous minute and the fingerprints of those devices that have been scanned and successfully fingerprinted. The device count is particularly valuable for crowd-monitoring operations and can also be used to develop predictive models. Meanwhile, the fingerprints serve to identify mobility patterns, enabling the creation of detailed mobility models.

The fingerprints obtained from the dataset offer a way to track near-anonymous devices and, by extension, near-anonymous users. It is possible to infer paths and itineraries by tracking a specific fingerprint from a start node, through one or more waypoints, and finally to an end node. As each node detects the same fingerprint at different times, it can piece together a likely travel route through a geographical area covered by station nodes, such as a city.

Name	Duration (mins)	Distinct MACs	Strategy used		# Fingerprints		Quantity
			Passive Listening	Scan Trigger	Enquiring	JustStore	
San Rocco	125	8263	1208	7086	157	22	179
Pub	23	532	123	416	17	12	29

Table 4.2: Results from the indoor/outdoor experiments. It shows the different metrics for the strategies of BLENDER, proving the effectiveness of the *ScanTrigger* strategy.

An example of path inference, derived from data collected during a test in Rome, is shown in Figure 4.7. In this instance, the same fingerprint was detected by three different stations located in the city center, near points of historical interest and public transportation stops, which served as reference points:

- Station 2001 at Colosseo subway (start point - 23 March 2022 10:12:00 GMT, timeslot 11)
- Station 2002 at via San Gregorio, near Arco di Costantino (waypoint - 23 March 2022 10:44:00 GMT, timeslot 61)
- Station 2003 at Circo Massimo tram stop (end point - 23 March 2022 10:57:00 GMT, timeslot 16)

The timeslots represent subintervals, approximately a quarter of a second in duration, during which the fingerprinted device was detected. Stations are georeferenced with latitude and longitude coordinates. To model the city of Rome and infer a walking path through various nodes, Python was used in conjunction with the OSMnx and NetworkX libraries, utilizing geographical data from Open Street Map. The inferred route was mapped based on actual streets and pedestrian paths.

In this example, the path starts at the Colosseo, where station 2001 was located, passes through station 2002 at San Gregorio Road, and ends at Circo Massimo, where station 2003 was positioned. The inferred path was visualized using Folium, showcasing the feasibility of reconstructing movement through sparse geographical fingerprinting data across widely spaced outdoor stations.

Two additional experiments were carried out to evaluate the performance of BLENDER.

1. **San Rocco experiment** — This test was conducted during San Rocco festival, in Trivigliano (FR), Italy. The experiment took place for a total duration of 125 minutes, during the evening. The three stations with IDs 1001,1002 and 1003 were equipped with portable UPS and deployed at the site, within a radius of approximately 60 meters in a stationary configuration. The three stations automatically collected the data presented in the quantitative indicators table.
2. **Pub experiment** — This test was conducted at a public pub in Rome, Italy, over a duration of approximately 23 minutes. A single station, powered by a portable source, was deployed in a fixed position at the site. Unlike the other two experiments, which focused on tracking devices and bypassing MAC address randomization, the goal of this test was to demonstrate the station's capability to count and differentiate between multiple devices present at a specific location and time.

Table 4.2 presents quantitative data collected from two experiments using two BLENDER station nodes. The table reveals a notable difference in the number of distinct MAC addresses detected

by the stations, which can be attributed to both the duration of the experiments and the environmental characteristics in which they were conducted. Specifically, the number of MAC addresses detected during the San Rocco experiment is approximately 15.5 times greater than the number detected in the Pub experiment, despite the collection time at San Rocco being only 5.43 times longer. This suggests that, under similar user density conditions, the stations in the San Rocco experiment were able to better utilize the full range of BLE compared to those in the Pub experiment. This discrepancy can be explained by the differing environments: the San Rocco stations operated outdoors in an open field, allowing for maximal exploitation of the BLE range, while the Pub experiment took place indoors, where walls and objects attenuated and reflected the BLE signals. As a result, the ability to detect distinct MAC addresses was reduced by about threefold in the indoor environment compared to the outdoor setting.

The data collected during the detection phase also revealed additional insights. When comparing MAC address discovery via *PassiveListening*, it becomes clear that the San Rocco experiment detected 1208 MAC addresses, while only 123 were detected in the Pub experiment—a difference of about 9.82 times. This ratio is nearly double the time difference between the two experiments. The disparity is even more pronounced for MAC addresses detected via the *ScanTrigger* method, where the San Rocco experiment identified 7986 addresses compared to just 416 in the Pub experiment, resulting in a ratio of approximately 17:1. The overall ratio between *ScanTrigger* and *PassiveListening* was 5.86:1 in the San Rocco experiment, while in the Pub experiment, it was 3.38:1.

This highlights that the impact of *ScanTrigger* relative to *PassiveListening* is greater in outdoor settings, while it diminishes in indoor environments for the same reasons. The considerable difference between the number of detections from *ScanTrigger* compared to *PassiveListening* confirms that *ScanTrigger* is an effective complementary strategy. As more BLE devices, particularly high-end smartphones, limit advertising due to privacy considerations, the importance of *ScanTrigger* for device discovery becomes even more evident.

In the first experiment, station 1001 collected a total of 179 fingerprints, with 157 obtained through *Enquire* (active scanning and fingerprinting using GATT) and 22 through *JustStore* (public/static MAC address fingerprinting). This results in an *Enquire*:*JustStore* ratio of approximately 7.13. In the Pub experiment, *Enquire* generated 17 fingerprints, while *JustStore* produced 12, giving a total of 29 fingerprints and an *Enquire*:*JustStore* ratio of about 1.41. These figures highlight a significant difference between the indoor and outdoor experiments in terms of fingerprint generation.

Regarding the calculated fingerprints, the system successfully identified identical fingerprints at different times, as expected. Table 4.3 highlights the occurrences of three fingerprints detected during the San Rocco experiment, clearly illustrating multiple detections at various instants and locations within the experiment scenario. From these records, the movement patterns of the same device can be inferred within the detection environment.

It is important to note that only the times are shown, specific to the San Rocco experiment, and for brevity, only short fingerprints (5 bytes) are displayed. Additionally, the last 3 bytes of the MAC addresses were obscured. One fingerprint, 6501C59023, corresponds to five different random MAC addresses frequently rotated by the same device. Another fingerprint, 844BA3ACF9, was derived from three random MAC addresses of a second device, although its rotation timing was slightly

non-compliant with the standard. Lastly, fingerprint 8A402A3139 was repeatedly computed from the same *random* MAC address of a device over extended periods, suggesting non-compliance with MAC address randomization, as the device did not rotate its MAC address.

Station ID	Latitude	Longitude	Time	Fingerprint	MAC
1001	41.776169	13.272443	21:43:31.723833	6501c59023	50:2E:02:**:**:**
1001	41.776169	13.272443	21:47:41.348958	6501c59023	50:5F:64:**:**:**
1001	41.776169	13.272443	21:49:43.832345	6501c59023	50:5F:64:**:**:**
1002	41.776489	13.272569	21:53:41.628960	6501c59023	50:5F:64:**:**:**
1003	41.776688	13.273054	21:57:34.623797	6501c59023	42:97:20:**:**:**
1002	41.776489	13.272569	22:24:12.894568	6501c59023	51:F9:FC:**:**:**
1002	41.776489	13.272569	22:23:41.056393	844ba3acf9	55:71:7C:**:**:**
1003	41.776688	13.273054	22:31:10.954223	6501c59023	51:F9:FC:**:**:**
1003	41.776688	13.273054	22:30:11.384897	6501c59023	5E:60:C9:**:**:**
1003	41.776688	13.273054	22:33:20.402627	6501c59023	5E:60:C9:**:**:**
1003	41.776688	13.273054	22:41:24.659739	8a402a3139	6D:69:41:**:**:**
1003	41.776688	13.273054	22:43:30.901356	8a402a3139	6D:69:41:**:**:**
1003	41.776688	13.273054	22:44:15.726762	8a402a3139	6D:69:41:**:**:**
1002	41.776489	13.272569	22:46:10.672153	8a402a3139	6D:69:41:**:**:**
1002	41.776489	13.272569	22:48:04.615016	8a402a3139	6D:69:41:**:**:**
1002	41.776489	13.272569	22:49:13.673665	8a402a3139	6D:69:41:**:**:**
1003	41.776688	13.273054	22:49:05.068207	844ba3acf9	56:82:B9:**:**:**
1002	41.776489	13.272569	22:50:31.020064	8a402a3139	6D:69:41:**:**:**
1003	41.776688	13.273054	22:52:20.865793	8a402a3139	6D:69:41:**:**:**
1003	41.776688	13.273054	22:57:40.434737	8a402a3139	6D:69:41:**:**:**
1001	41.776169	13.272443	23:10:33.528203	844ba3acf9	59:21:76:**:**:**

Table 4.3: San Rocco data samples. MAC addresses have been partially obscured.

4.5 Conclusions

In conclusion, the BLENDER system demonstrates the limitations of MAC address randomization as a privacy measure in BLE devices, revealing that this mechanism alone is insufficient to prevent device tracking. By employing a combination of passive listening and active scanning strategies, BLENDER effectively bypasses this security feature, enabling the discovery, identification, and tracking of devices even when their MAC addresses are frequently rotated.

The *PassiveListening* strategy captures advertising packets and extracts MAC addresses, but the increasing use of randomization in high-end devices has reduced the efficacy of this approach alone. To counter this, BLENDER introduces the innovative *ScanTrigger* method, which detects devices that do not actively advertise but are scanning in the background. This strategy significantly boosts detection rates, particularly in environments with high interference, showing that even devices attempting to conceal their presence through randomization can still be exposed.

Additionally, BLENDER utilizes the Enquiring strategy to actively fingerprint devices by retrieving static data from the GATT, allowing for unique device identification despite the randomized MAC addresses. The *JustStore* approach complements this by focusing on devices that continue

to use static MAC addresses, such as smart bands and fitness trackers, highlighting another vulnerability in the randomization strategy.

The experiments in both outdoor and indoor environments further illustrate how MAC address randomization fails to provide full protection. In outdoor scenarios, the system effectively utilizes the broader detection range, while in indoor environments, signal attenuation affects detection, but the system still proves capable of identifying and tracking devices.

BLENDER conclusively demonstrates that MAC address randomization, while offering some level of privacy, is not enough to protect devices from tracking. The system's combination of passive and active scanning methods, alongside robust fingerprinting techniques, reveals significant weaknesses in randomization and underscores the need for more comprehensive privacy measures in BLE networks. Through these methods, BLENDER provides clear insights into the movements and presence of BLE devices, even in environments where privacy measures like MAC randomization are in place.

Chapter 5

Decentralize LPWANs

5.1 Low-Power Wide-Area Networks: An Overview

In recent years, the demand for wide-area communication technologies that support low-power, low-data-rate devices has grown significantly due to the rapid proliferation of the Internet of Things (IoT). This surge in connected devices, ranging from smart sensors to remote monitoring systems, has fueled the need for communication networks that can handle the unique challenges of IoT applications. Unlike traditional internet-connected devices, which often rely on short-range networks like Wi-Fi or Bluetooth (BT), IoT devices frequently operate in environments where power availability is limited, and battery life is a critical factor. In response to this, Low-Power Wide-Area Networks (LPWANs) have emerged as one of the most promising solutions to address the specialized needs of IoT applications, particularly in scenarios where devices are distributed over vast geographic areas and need to operate on minimal power resources. The concept behind LPWANs lies in the ability to strike a delicate balance between power efficiency, long-range communication, and low data throughput. While cellular networks, such as LTE and 5G, are designed to prioritize high-speed data transmission and intensive connectivity demands, they are typically overkill for many IoT applications. Cellular networks are energy-intensive, making them unsuitable for devices that are expected to last for years on a small battery. LPWANs, by contrast, are tailored specifically for applications where low-bandwidth communication is sufficient, but where devices must operate efficiently and autonomously over extended periods of time. By focusing on energy conservation and extended device lifetimes, LPWANs have become a cornerstone technology for sectors like smart agriculture, industrial automation, environmental monitoring, and smart city infrastructure.

The origins of LPWANs can be traced back to the early 2000s [44, 45], a period during which the limitations of existing wireless protocols, such as Wi-Fi and BT, became increasingly apparent for emerging IoT use cases. While these technologies were highly effective for short-range, high-throughput applications, they fell short in terms of range and power efficiency, two critical factors for IoT devices that needed to operate remotely, often in inaccessible locations. As a result, the early 2000s saw growing interest in developing a new class of networks specifically tailored to the requirements of the IoT landscape. Key industries, particularly those involved in environmental monitoring, smart farming, and industrial systems, began to identify a pressing need for wireless networks that could connect devices spread across large areas, such as farms, forests, and factories, while ensuring that those devices could function for years without the need for frequent maintenance

or battery replacement.

One of the primary drivers behind the development of LPWANs was the realization that many IoT devices did not require constant, high-speed data transmission. Instead, these devices often needed to send small packets of information, such as sensor readings or status updates, intermittently over long periods. For example, a soil moisture sensor deployed in an agricultural field might only need to transmit data a few times a day, yet it must be able to maintain connectivity to the network across large distances and operate for years on a single battery. This is where LPWANs excel: they provide a low-cost, low-power communication solution that can transmit small amounts of data over large distances, making them ideal for these types of applications.

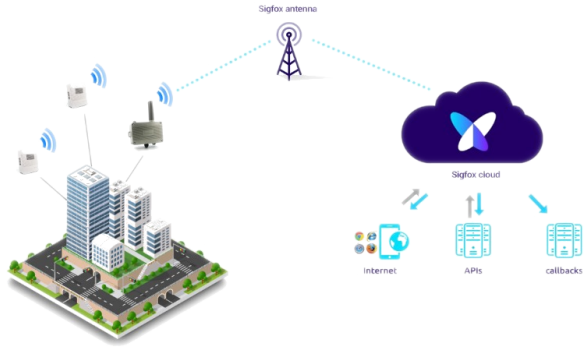
Several LPWAN technologies have emerged as leaders in this space, including LoRaWAN [21], SigFox [46], and Narrowband IoT (NB-IoT) [47]. These technologies are designed with similar goals, extending battery life and maintaining reliable communication over large areas, but they differ in terms of how they achieve these goals, their technical characteristics, and the specific use cases for which they are best suited:

1. **LoRaWAN** is one of the most widely adopted LPWAN technologies, primarily due to its flexibility, scalability, and open nature. LoRaWAN leverages Chirp Spread Spectrum (CSS) modulation, which allows it to achieve long-range communication even in environments with significant interference. Key benefits of LoRaWAN are its ability to support different classes of bi-directional communication, together with its open protocol. The wide variety of hardware have made it the first choice for smart city projects, agriculture, industrial automation, and environmental monitoring.

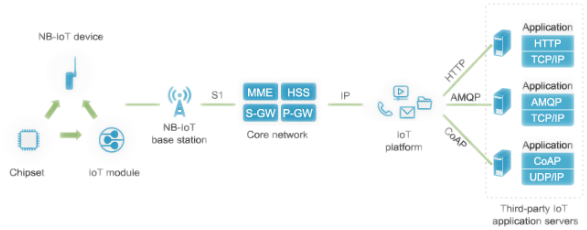
2. **SigFox**, another prominent LPWAN technology, takes a different approach. Operating as a proprietary, ultra-narrowband technology, SigFox is designed for low-throughput applications where devices need to send small bursts of data. SigFox operates on its own network of base stations, which are deployed and managed by the company itself, providing global coverage in certain regions. This proprietary model allows SigFox to offer a highly controlled and optimized service for specific IoT applications, such as asset tracking or remote monitoring. However, its limitations in data throughput and its dependence on SigFox-operated infrastructure can make it less suitable for more complex or dynamic applications where higher data rates or bi-directional communication are required.

3. **NB-IoT** represents a cellular-based LPWAN solution developed as part of the 3GPP standards for LTE. Unlike the others, NB-IoT operates in licensed spectrum, leveraging existing cellular networks to provide extended coverage for IoT devices. NB-IoT is particularly suited for applications that require a guaranteed Quality of Service (QoS) and deep indoor penetration, such as smart meters or building automation systems. While its reliance on cellular infrastructure means that NB-IoT can offer highly reliable and interference-free communication, it also comes with higher deployment and operational costs compared to unlicensed LPWAN technologies like LoRaWAN and SigFox.

As the IoT ecosystem continues to grow, LPWANs have solidified their role as essential communication technologies for enabling smart, connected environments. Their ability to combine



(a) SigFox follows a star topology, where end devices send messages to SigFox base stations, which relay them to a centralized cloud for processing. All network functions, including data handling and device management, are managed centrally in the SigFox cloud.



(b) NB-IoT uses a star topology, where end devices connect directly to cellular base stations, which forward data to the core network. By leveraging existing cellular infrastructure, NB-IoT provides deep coverage and ensures reliable communication with guaranteed QoS.

long-range connectivity with low power consumption makes them the preferred choice for a wide array of IoT applications, particularly those that demand long-term, autonomous operation in remote or challenging environments.

5.1.1 Centralized LPWANs

LPWANs generally follow a star or star-of-stars topology, which differentiates them from mesh-based personal networks like Zigbee or BT Mesh. In a basic star topology, End Devicess (EDs) communicate directly with a central base station or gateway, which then forwards the data to a central authority for processing. This simplicity is one of the reasons LPWANs can maintain low power consumption, as EDs are only required to transmit over long distances to the nearest gateway, without the need to forward messages to other devices.

LoRaWAN, as described in Chap. 3, operates in a star-of-stars topology (Fig. 3.2), where EDs send uplink messages to nearby Gateways (GWs), which then forward the data to a centralized Network Server (NS). The NS is responsible for critical network management tasks such as message deduplication, authentication, encryption, and routing data to the Application Server (AS). Unlike mesh networks, where devices pass messages through one another to extend range and coverage, LPWANs prioritize direct communication between devices and the central infrastructure to minimize power usage.

SigFox too operates with a star-of-stars topology (Fig. 5.1a), where EDs, also known as nodes or IoT devices, communicate directly with SigFox-operated base stations. These base stations are strategically distributed across large geographic areas and serve as intermediaries that receive messages from the devices and forward them to SigFox's centralized cloud infrastructure for processing. One of the defining characteristics of SigFox is its primary focus is on uplink communication, where devices send small bursts of data to the network. Downlink communications, while possible, are limited. SigFox's architecture is highly centralized, with base stations merely relaying messages to the cloud, where all the data processing, message deduplication, and device management occur. This centralized backend approach allows SigFox to maintain a lightweight network structure while providing global coverage. Unlike traditional networks, devices can connect to any SigFox base station within range, enabling seamless roaming without the need for complex handover mechanisms.

NB-IoT also employs a star topology, as shown in Fig. 5.1b, but it leverages the existing

infrastructure of cellular networks, such as LTE or 5G. IoT devices communicate directly with cellular base stations, which relay their messages to the core network for further processing. One of the key advantages of NB-IoT is its use of licensed spectrum, which ensures guaranteed QoS and minimal interference, making it suitable for critical applications like smart metering and healthcare devices. Additionally, NB-IoT is designed to provide deep coverage, particularly for devices located indoors or in difficult-to-reach areas, such as basements or underground facilities. It achieves this through the use of narrowband channels, which improve signal penetration and extend range. Unlike SigFox, NB-IoT fully supports two-way communication, allowing devices not only to send data to the network but also to receive commands or updates from it, making it ideal for real-time control applications.

A centralized architecture streamlines network management by consolidating control and data processing in a single location, such as the Network Server for LoRaWAN, the SigFox Cloud, or the core network in NB-IoT. This design simplifies monitoring, maintenance, and updates, minimizing the complexity associated with distributed systems. It also improves scalability, as centralized networks can manage a large number of devices using powerful centralized servers or cloud infrastructure. Furthermore, centralization strengthens security by enabling easier implementation of security protocols and more effective threat monitoring, simplifying the network remains robust and secure.

5.2 Centralization issues in LPWANs

As introduced in Sec. 3.2.7, while centralized architectures in LPWAN networks like LoRaWAN, SigFox, and NB-IoT offer simplicity and ease of management, they present several critical issues that can undermine the effectiveness and reliability of the network.

Single Point of Failure

One of the most significant vulnerabilities in centralized LPWAN architectures is the risk of a Single Point of Failure (SPoF) [48, 49]. In LPWANs, all communication and control functions are routed through a central entity, such as a NS or cloud infrastructure. This setup means that the entire network depends on the availability, integrity, and security of this single component. If the central server or infrastructure experiences a failure, whether due to a hardware malfunction, software error, or cyberattack, the entire network can become inoperable, disrupting communication between devices and halting critical IoT operations.

The consequences of such a failure can be severe, particularly in use cases that involve time-sensitive or mission-critical applications, such as industrial automation, healthcare monitoring, or smart cities. A SPoF can lead to prolonged downtime, loss of data, or compromised services, all of which can have significant financial and operational repercussions. Moreover, the time and resources required to restore the central entity can exacerbate the disruption, particularly if the failure occurs in a remote or hard-to-access location.

In addition to outages caused by internal failures, centralized architectures are also more vulnerable to external threats [50]. A targeted cyberattack, such as a Distributed Denial-of-Service (DDoS) attack or a data breach, can compromise the central server, effectively paralyzing the entire network. The centralized nature of the architecture makes it an attractive target for attackers,

as compromising the central node gives them control over the entire system. In these cases, the inability to operate independently of the central infrastructure leaves the network highly exposed to both intentional and unintentional disruptions. Furthermore, the centralization of control and data processing in one entity can also lead to performance degradation as the network scales. As the number of connected devices grows, the central server may struggle to handle the increased load, leading to bottlenecks, higher latency, and reduced throughput. This performance limitation further compounds the risk of SPoF, as the network becomes increasingly reliant on the capacity and robustness of the central node.

Data Security

The centralization of data storage in LPWAN networks presents another significant issue regarding the operational and security risks associated with a SPoF. In centralized architectures, all data transmitted by end devices is typically stored and processed in a single, centralized location, such as a cloud server, introducing several challenges. First, it makes the system vulnerable to data breaches, as attackers who compromise the central server can gain access to large volumes of sensitive information in one strike [51]. The concentration of data in a single location also increases the risk of loss in the event of a hardware failure other unforeseen incidents that could corrupt or destroy the stored data. Additionally, centralized data storage raises privacy concerns [52], particularly in applications where personal or sensitive information is transmitted, such as healthcare or smart home devices. Users and organizations may have limited control over how their data is stored, managed, and protected, making it difficult to ensure compliance with data privacy regulations. These issues highlight the need for more distributed or decentralized data storage solutions, which could offer greater resilience, security, and privacy by spreading data across multiple nodes, reducing the impact of a single failure or breach.

Trust

A additional critical issue with centralized LPWAN architectures is the inherent need for trust in the central authority that manages the network [53]. In centralized systems, users must place a great deal of trust in the central entity to securely manage data, maintain network availability, and uphold privacy standards. This centralized control gives the managing authority full access to sensitive data and the ability to influence or restrict network operations. If this entity is compromised or behaves maliciously, either through negligence or intentional actions, the entire network could be at risk. Moreover, users and organizations often have little visibility or control over how their data is handled, processed, or shared, especially when it is stored in centralized servers managed by third-party providers. This lack of transparency can lead to concerns about data misuse or regulatory non-compliance, especially in industries where data protection is crucial, such as healthcare or finance. The need for trust in centralized architecture therefore creates a single point of vulnerability, as users are dependent on the central authority's ability to safeguard the network and uphold its commitments to security and privacy.

5.3 Decentralize LPWANs - State of the Art

The issues associated with centralized LPWAN architectures have prompted researchers and industry practitioners to explore alternative approaches that can address the limitations of SPoF, data security, and trust. One promising solution that has gained significant attention is the use of blockchain technology to decentralize key components of LPWAN networks, such as the NS, Join Server (JS), and data storage. By leveraging blockchain's distributed ledger, consensus mechanisms, and smart contracts, researchers have proposed novel architectures that distribute control, data processing, and security functions across multiple nodes, reducing the risks associated with centralization. These decentralized LPWAN systems offer several advantages, including improved fault tolerance, enhanced security, and increased trust among network participants.

LoRaWAN has become a central focus in the development of IoT due to its open architecture, flexibility in meeting application needs and low-cost deployment. These characteristics have driven widespread adoption, positioning LoRaWAN as a key enabler of IoT growth. As a result, it has become a significant subject of research, particularly in addressing challenges related to security and scalability.

Below, we summarize the state of the art based on the referenced works, which investigate using blockchain to create decentralized and more secure LPWAN systems.

5.3.1 Mesh Networks of Gateways and Servers Using Blockchain

The centralization of gateways and NSs in conventional LoRaWAN architectures creates potential SPoFs and increases the network's vulnerability to attacks and failures. Researchers like Bezahaf et al. [54] and Durand et al. [55] have explored solutions based on blockchain to create decentralized mesh networks of gateways.

Bezahaf et al. [54] developed BcWAN, a federated blockchain-based network that enables gateways to be shared among different LoRaWAN operators, effectively reducing dependency on any single entity and promoting resource sharing. This solution enhances scalability and reliability by using blockchain to manage access control and gateway operations.

Similarly, Durand et al. [55] proposed a decentralized LPWAN infrastructure using blockchain and digital signatures. By introducing blockchain into the network infrastructure, this work enables peer-to-peer interactions among gateways and servers, ensuring secure data handling, verifiability, and distributed trust.

These efforts significantly enhance the fault tolerance and security of LoRaWAN systems by distributing responsibilities across multiple entities, reducing the likelihood of failure or attack on a single gateway or server.

5.3.2 Decentralization of the Key Management Using Blockchain

The JS is responsible for handling the Over-The-Air Activation (OTAA) process in LoRaWAN, and its centralized nature makes it a critical SPoF. To mitigate this risk, researchers like Ribeiro et al. [56] and Tan et al. [57] proposed blockchain-based architectures that decentralize the JS's duties.

Ribeiro et al. [56] introduced a secure and fault-tolerant architecture that distributes the key management responsibilities of the JS across a blockchain network. This approach uses smart

contracts to ensure that no single entity has full control over key management, enhancing the network's resilience.

Tan et al. [57] designed a blockchain-based key management scheme for LoRaWAN, which also leverages blockchain to distribute the JS's responsibilities. Their solution shortens the network access time and prevents attacks on the JS by using a permissioned blockchain that only allows authorized entities to read or write data.

Both solutions increase security by preventing attacks that could compromise the entire network, such as Denial-of-Service (DoS) attacks targeting the JS.

5.3.3 Enhancing Network Server Security with Blockchain

In addition to the JS, the NS in LoRaWAN is also a potential point of failure and attack. Lin et al. [58] proposed the use of a public blockchain to create a mesh network of NSs, which enhances trust and reliability by allowing multiple servers to share and validate network operations. This solution focuses on decentralizing the NS, ensuring that even if one server is compromised, others can continue to provide network services, thereby improving the overall availability of the LoRaWAN system.

5.3.4 Distributing Workload to the Edge Using Blockchain

While decentralizing key network components improves security, Hou et al. [59] and Wei et al. [60] have explored using blockchain to distribute workloads to edge nodes. This approach leverages edge computing to improve the performance of the LoRaWAN network, particularly in handling time-critical data. Hou et al. [59] proposed *HyperLoRa*, a blockchain-enabled LoRaWAN system with edge computing capabilities. Their design allows LoRaWAN gateways to process some network tasks traditionally handled by the central cloud, reducing latency and improving efficiency. Although this solution distributes some of the workload, it does not fully eliminate the centralized NS, meaning some vulnerabilities remain. Wei et al. [60] presented a DAG-based LoRaWAN system, which uses a Directed Acyclic Graph (DAG) instead of a traditional blockchain to improve the system's throughput and latency. The DAG structure allows for asynchronous consensus, making it more scalable for large networks. However, while this system improves performance, it also fails to fully eliminate reliance on a central NS.

5.4 Blockchain to Decentralize LPWANS

The research efforts highlighted in this section demonstrate that blockchain presents a feasible approach to decentralizing critical components of LoRaWAN systems (Fig. 5.2), effectively mitigating the security risks posed by SPoFs. The implementation of blockchain-based key management, decentralized Join Servers, and distributed edge computing offers substantial potential for developing more resilient and scalable LPWAN infrastructures. However, despite the security enhancements and reduced reliance on centralized servers, to the best of our knowledge, no existing work fully addresses the elimination of SPoFs while also removing the need for trust.

Blockchain technology plays a pivotal role in decentralizing traditionally centralized LPWAN architectures, such as LoRaWAN, SigFox, and NB-IoT. The inherent qualities of blockchain, like

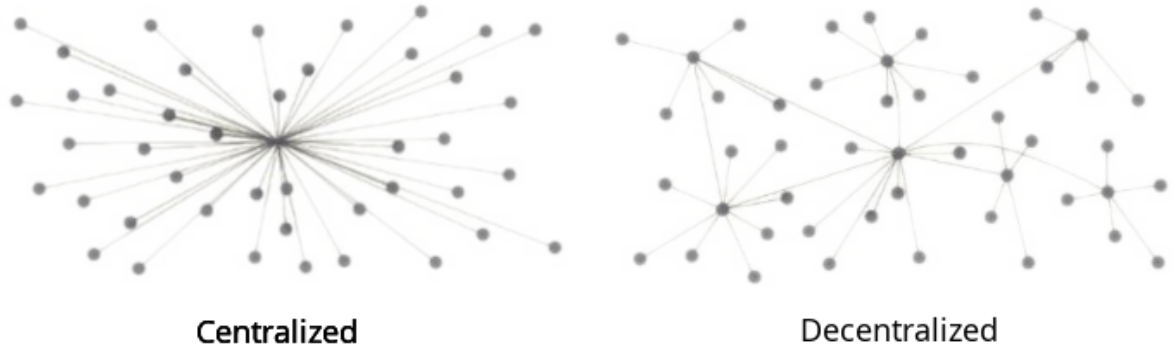


Figure 5.2: Centralized vs Decentralized Network. In a centralized network, all communication and control functions are routed through a single entity, creating a SPoF and increasing the network's vulnerability to attacks. In a decentralized network, control, data processing, and security functions are distributed across multiple nodes, reinforcing the network.

distributed ledger systems, consensus mechanisms, and decentralized trust, address many of the challenges associated with centralization, such as SPoFs, security vulnerabilities, and the need for trust in a single authority.

5.4.1 What is a Blockchain

A blockchain is a decentralized, distributed ledger technology designed to securely record and verify transactions without the need for a central authority or intermediary. It works by maintaining a growing list of records (blocks) that are linked using cryptography. Each block contains a cryptographic hash of the previous block, a timestamp, and transaction data, ensuring that once data is added, it becomes part of an immutable and transparent chain of records. This structure, as shown in Fig. 5.3 makes tampering with the data nearly impossible without altering the entire chain, which would require the consensus of the majority of participants.

The concept of blockchain was originally developed as the underlying technology for Bitcoin [61], solving the problem of trust in decentralized digital transactions. Before blockchain, transactions required a trusted intermediary, such as a bank or payment processor, to verify the validity of transactions and ensure that the same digital asset was not spent more than once (the double-spending problem). Blockchain's decentralized nature removes this need by allowing all participants in the network to agree on the validity of transactions through a consensus mechanism.

Consensus mechanisms, are the cornerstone of blockchain technology, ensuring that all participants in the network agree on the current state of the ledger, which contains all validated transactions. In decentralized systems like blockchain, there is no central authority to verify transactions or maintain data integrity. Instead, consensus is achieved through mechanisms that ensure trust and coordination among independent, often anonymous and untrusted, participants.

Some noteworthy examples are shown in Tab. 5.1. In Proof-of-Work (PoW), used by Bitcoin, nodes (miners) solve complex puzzles to add new blocks to the blockchain. This process is highly energy-intensive, but it secures the network by making it prohibitively costly for attackers to alter the blockchain, as they would need to control over 51% of the network's computational power. In contrast, Proof-of-Stake (PoS) is more energy-efficient. Validators are chosen to propose and validate blocks based on the amount of cryptocurrency they "stake" as collateral. If they act maliciously, they

Consensus Mechanism	Description	Resource Intensity	Typical Use Case
Proof of Work (PoW)	Miners solve complex puzzles to validate transactions.	Energy-intensive	Bitcoin, Ethereum (pre-2.0)
Proof of Stake (PoS)	Validators are chosen based on the number of tokens they "stake".	Economically intensive, energy-efficient	Ethereum 2.0, Cardano
Delegated Proof of Stake (DPoS)	Token holders elect a small group of validators to create blocks.	Moderately centralized, energy-efficient	EOS, TRON
Byzantine Fault Tolerance (BFT)	Nodes reach consensus despite some acting maliciously or failing.	Memory-intensive	Hyperledger Fabric, Tendermint (Cosmos)
Proof of Authority (PoA)	A limited number of trusted validators are selected to secure the network.	Energy-efficient, requires trust in validators	VeChain, private/ permissioned blockchains

Table 5.1: Comparison of consensus mechanisms, highlighting use cases and key features of the protocol in terms of use of energy, memory and trust

risk losing their stake, aligning their incentives with network security. PoS eliminates the need for the energy consumption associated with PoW while still maintaining the integrity of the network. Both PoW and PoS solve the problem of double-spending, ensuring that once a transaction is validated and recorded, it cannot be altered, maintaining the trust and immutability of the blockchain.

One of the critical aspects of consensus mechanisms, particularly in decentralized networks, is their ability to handle Byzantine Faults [62]. These occur when some nodes in the network act maliciously or provide incorrect information. Consensus algorithms, like Byzantine-Fault Tolerant (BFT), ensure that the network can still reach agreement even if a portion of the nodes behave dishonestly. For example, a majority of honest nodes can collectively validate the correct state of the blockchain, rendering any misleading information from rogue nodes ineffective. This makes the blockchain resilient to internal attacks.

Blockchain, through consensus, solves the core problems of trust and data integrity in a decentralized system. Whether through the energy-intensive PoW, the economically aligned PoS, or fault-tolerant BFT algorithms, blockchain consensus ensures that all participants can trust the system's outputs without relying on a central intermediary. This resilience, security, and decentralized trust make blockchain the most suitable solution for decentralized applications, where network security and fault tolerance are paramount.

Blockchain solves several key problems:

- **Trust**: It removes the need for a trusted central authority by distributing trust among the network participants;
- **Transparency**: All transactions are visible on a public ledger, allowing anyone to audit the network and verify transactions;
- **Security**: The cryptographic linking of blocks ensures that altering the data in one block

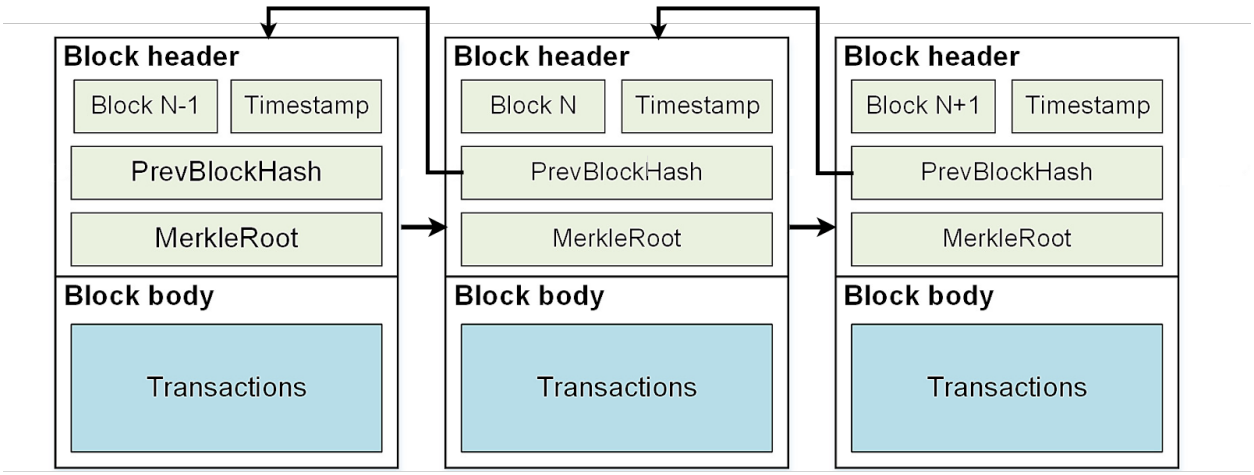


Figure 5.3: A blockchain is composed of a list of blocks, linked by storing in each block the hash, the fingerprint of the previous block. This ensures the immutability of the content stored on the blockchain.

would require altering all subsequent blocks, making the system highly resistant to tampering;

By decentralizing control and securing data in an immutable ledger, blockchain has applications beyond cryptocurrencies, such as in supply chain management, healthcare, finance, and decentralized networks like DeLoRaN.

5.4.2 Benefits of a Blockchain

In a centralized LPWAN the data processing unit forms the backbone of the network, but this concentration of control introduces significant risks. If the core functionalities are compromised, the entire network could be rendered inoperable, halting communications and potentially causing large-scale data loss. By integrating blockchain technology, network functions such as authentication, key management, and data processing can be distributed across multiple nodes, ensuring that no SPoF exists. This distribution significantly improves fault tolerance, as the network can continue to operate even if individual nodes or servers fail. Blockchain's decentralized architecture ensures that decision-making and data handling are spread across the network, preventing any single node from becoming a critical failure point.

Moreover, blockchain enhances security in LPWANs by making it much harder for malicious actors to compromise the network. The transparency and immutability of blockchain ensure that once data is recorded in the distributed ledger, it cannot be altered or tampered with without consensus from the entire network. This property is particularly valuable for IoT applications, where data integrity is critical. Additionally, blockchain-based consensus mechanisms, such as Practical Byzantine-Fault Tolerant (PBFT), ensure that only authenticated nodes can participate in the network's operations, making it difficult for unauthorized entities to launch attacks like DDoS or data breaches. This is particularly useful in LPWANs where devices are often deployed in large numbers, making them vulnerable to security threats.

Blockchain also addresses the trust issues inherent in centralized LPWANs. In traditional architectures, users must trust the central authority to manage and store their data securely. However, in a blockchain-based system, trust is decentralized, and data is stored in a distributed manner across multiple nodes. This decentralization reduces the need to rely on a single authority, allowing net-

work participants to have greater control over their data. Blockchain also provides transparency, as all network transactions and data exchanges are recorded on a public or permissioned ledger, which can be audited by authorized entities. This transparency builds trust among network participants and ensures compliance with data privacy regulations, which is crucial for industries like healthcare or finance, where sensitive information is frequently transmitted.

Additionally, blockchain can enhance the scalability of LPWAN networks. In centralized systems, as the number of connected devices grows, the central server can become overwhelmed, leading to performance bottlenecks. Blockchain-based LPWANs, on the other hand, can distribute the workload across multiple nodes, allowing for greater scalability. Each node in the blockchain network can process a portion of the network's transactions, reducing the burden on any single server and improving the overall performance of the network. This is particularly useful for large-scale IoT deployments, such as smart cities or industrial automation, where thousands or even millions of devices need to be connected and managed simultaneously.

Lastly, blockchain enables the implementation of smart contracts in LPWAN networks. Smart contracts are self-executing contracts with the terms of the agreement directly written into code. In the context of LPWANs, smart contracts can automate tasks like device onboarding, resource allocation, and data sharing, making the network more efficient and reducing the need for human intervention. For example, a smart contract could automatically verify a device's credentials and grant it access to the network, streamlining the authentication process and reducing the time and resources needed for network management.

Blockchain's decentralized architecture, enhanced security features, transparency, and scalability make it a key technology for distributing centralized LPWANs. By eliminating SPoF, reducing the need for trust in a central authority, and enabling more efficient network management, blockchain offers a robust solution for the challenges faced by traditional LPWAN architectures. As IoT continues to expand, integrating distributed LPWANs will be crucial for creating resilient, scalable, and secure networks capable of supporting the next generation of connected devices.

Chapter 6

DeLoRaN - Decentralized LoRaWAN Network

6.1 Introduction

Building on the advantages of decentralizing Low-Power Wide-Area Networks (LPWANs), DeLoRaN represents a novel solution specifically designed to overcome the vulnerabilities of LoRaWAN's centralized architecture. While LoRaWAN is one of the most widely adopted LPWAN technologies, its reliance on a centralized Network Server (NS) and Join Server (JS) presents several critical limitations, such as a Single Point of Failure (SPoF), security risks, and performance bottlenecks in large-scale deployments.

DeLoRaN, presented in [63], addresses these limitations by decentralizing the control mechanisms in LoRaWAN, distributing network tasks such as authentication, data handling, and key management across multiple nodes using blockchain technology. By doing so, DeLoRaN eliminates the reliance on a central authority, thereby enhancing the network's resilience and security. In a traditional LoRaWAN setup, the NS is a key component responsible for managing all network operations, but its central role makes it vulnerable to failure or attack. DeLoRaN solves this by introducing a decentralized consensus mechanism, ensuring that no single node has control over the network, which increases fault tolerance and reduces the impact of node failures. Additionally, DeLoRaN leverages the inherent properties of blockchain, such as immutability and transparency, to enhance trust within the network. Instead of relying on a central authority, all network participants can verify transactions and ensure the integrity of the system through the distributed ledger. This decentralized trust model is particularly useful for IoT applications that require high levels of data integrity and privacy, as it reduces the risk of data tampering or unauthorized access.

Scalability is another key benefit of DeLoRaN. As the number of devices in an IoT network grows, traditional LoRaWAN architectures may struggle to manage the increased load on a centralized server. DeLoRaN's decentralized approach distributes network operations across multiple nodes, allowing for more efficient handling of large-scale deployments. This is especially advantageous for use cases like smart cities or industrial IoT, where thousands or even millions of devices need to communicate reliably and securely over long distances.

By integrating blockchain into LoRaWAN, DeLoRaN creates a more robust, scalable, and secure solution that mitigates many of the challenges associated with centralized LPWAN architectures.

This decentralized approach allows for more reliable and resilient IoT networks, better suited to the demands of modern applications that require long-term autonomous operation and data security.

In addition to its technical advantages, DeLoRaN is particularly well-suited for a multi-tenant environment, where different operators share the same network infrastructure. In traditional setups, each operator typically maintains its own independent network, which can lead to inefficiencies, redundancy, and difficulties in coordinating activities between different stakeholders, even causing disruption because of inter-network interferences [64]. However, by leveraging blockchain in DeLoRaN, multiple operators can participate in a shared network without fully relying on or needing to trust one another.

6.2 From LoRaWAN to DeLoRaN

Switching from the traditional centralized LoRaWAN architecture to the decentralized DeLoRaN model involves a fundamental restructuring of the network's core operations. In the centralized LoRaWAN framework, the NS serves as the central point for all data handling, authentication, and key management processes. This structure, while efficient due to its semplicity, becomes a bottleneck as the network grows, leading to potential single points of failure, performance limitations, and security vulnerabilities, as the entire system relies heavily on the availability and integrity of the NS.

DeLoRaN addresses these limitations by decentralizing the NS and other critical components, like the JS and the storage, using blockchain technology, as represented in Fig. 6.1. In this new architecture, tasks such as authentication, key management, and data routing are distributed across multiple nodes rather than being handled by a central authority. This distribution reduces the risk associated with SPoFs, if one node or component fails, the network continues to function smoothly, as other nodes can take over the compromised node's responsibilities. Blockchain's decentralized consensus mechanism ensures that all nodes in the network agree on the network state and operations, further enhancing the robustness and security of the network.

By moving to a decentralized architecture, DeLoRaN allows for greater scalability. Instead of relying on a single NS that can become overwhelmed with a growing number of connected devices, network control is distributed, enabling the network to accommodate more devices without performance degradation. The distributed ledger provided by blockchain also adds an extra layer of security, ensuring that data integrity is maintained, and malicious actors cannot easily alter network operations.

This switch not only improves resilience and scalability but also enhances trust within the network. In centralized LoRaWAN, users must trust the NS to securely manage their data, which can raise privacy concerns. DeLoRaN reduces this reliance on a single authority by using blockchain to create a transparent and immutable record of all transactions, giving users confidence in the network's security and data management practices. As a result, DeLoRaN represents a significant evolution in LPWAN technology, providing a more secure, scalable, and resilient alternative to the traditional centralized LoRaWAN architecture.

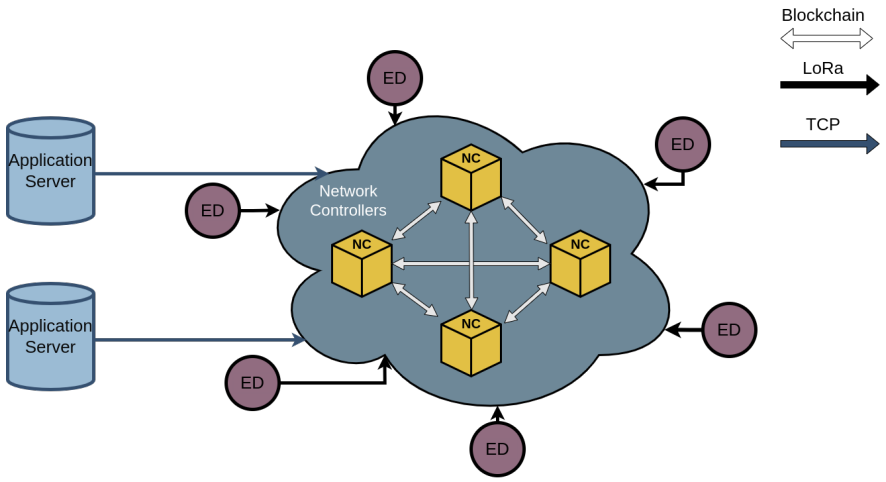


Figure 6.1: In DeLoRaN, multiple Network Controllers (NCs) are connected via blockchain, distributing control tasks across several nodes, improving fault tolerance and scalability. The End Devicess (EDs) communicate with the network through gateways, which relay messages to the NCs. The data is then processed and forwarded to the Application Servers (ASs) for further handling. Blockchain ensures secure and transparent coordination between NCs, while Long Range (LoRa) and TCP protocols handle communication between devices and the network.

DeLoRaN Advantages

DeLoRaN offers several key advantages over traditional LoRaWAN, specifically addressing critical limitations of the latter. One of the main improvements is enhanced resilience.

- **Enhanced Resilience**: In LoRaWAN, the NS acts as a central point for all network tasks, making the system vulnerable to SPoFs. DeLoRaN mitigates this by distributing control and data processing across multiple NCs, reducing the impact of node failures or attacks;
- **Scalability**: Traditional LoRaWAN networks may struggle with performance degradation as the number of connected devices increases due to the load on a single NS. DeLoRaN distributes the load across multiple controllers, ensuring that the network can scale to accommodate large Internet of Things (IoT) deployments without experiencing bottlenecks;
- **Security**: DeLoRaN leverages blockchain technology to introduce decentralized trust mechanisms. This eliminates the need to rely on a single trusted entity (like the NS in LoRaWAN) for tasks like key management and data verification. The immutable and transparent nature of blockchain ensures that all transactions and interactions between NCs are secure and verifiable, protecting against attacks such as man-in-the-middle and data tampering;
- **Data Integrity and Privacy**: Since blockchain is used to store critical network interactions, the transparency and immutability of the ledger reduce the risks of data manipulation. This is particularly beneficial for IoT applications that handle sensitive data, as it improves compliance with privacy regulations and provides users with greater confidence in the network's security.

DeLoRaN tradeoffs

Decentralization in DeLoRaN inevitably introduces trade-offs when compared to centralized approaches, particularly regarding complexity, maintenance, and resource requirements. One of the most prominent challenges is the increased infrastructure complexity. Managing a decentralized network requires more sophisticated mechanisms for ensuring that all nodes are properly synchronized and that consensus is achieved without a central authority. This added complexity demands higher technical expertise and can lead to more costly and labor-intensive maintenance compared to the relatively straightforward management of a centralized system.

Additionally, while decentralization eliminates SPoF, it introduces the challenge of coordination among nodes. Ensuring that multiple DeLoRaN NCs remain synchronized and act in unison can introduce coordination overhead. This requires more advanced monitoring and error-handling mechanisms, particularly in cases where nodes might fail or provide conflicting data. Such scenarios necessitate the use of consensus protocols, which, while improving security, add a layer of operational overhead.

Furthermore, scalability challenges arise as the number of nodes increases. While decentralization theoretically supports better scaling, a large number of independent nodes in a mesh network can lead to exponential increase in bandwidth and communication overhead as these nodes must constantly verify transactions and communicate with each other. This can strain network resources, particularly in environments with limited bandwidth or computational power, which is often a concern in IoT deployments like LoRaWAN.

Another important trade-off is resource consumption. While a centralized system consolidates processing power and network management, decentralization requires that every node perform its own set of tasks, such as validating transactions or storing parts of the ledger. This distributes the resource burden across the network, but it leads to an increased workload at the edge, particularly in resource-constrained environments. For instance, each node in a decentralized system like DeLoRaN must maintain a copy of the blockchain, which can result in higher storage and processing requirements compared to a single centralized server handling all data.

While decentralization offers key benefits such as fault tolerance, security, and autonomy, it comes at the cost of increased complexity, resource demands, and coordination challenges. These trade-offs must be carefully considered when deciding between centralized and decentralized architectures, particularly in environments where simplicity and ease of maintenance are priorities.

6.3 DeLoRaN on Blockchain

Blockchain plays a central role in DeLoRaN by addressing key challenges that arise in decentralized networks. In a system where multiple NCs need to operate simultaneously, blockchain ensures that state changes and transactions are securely validated and stored without relying on a centralized SPoF. By distributing control and implementing a decentralized trust model, DeLoRaN mitigates the risks associated with compromised or malicious nodes, using consensus mechanisms to maintain network integrity.

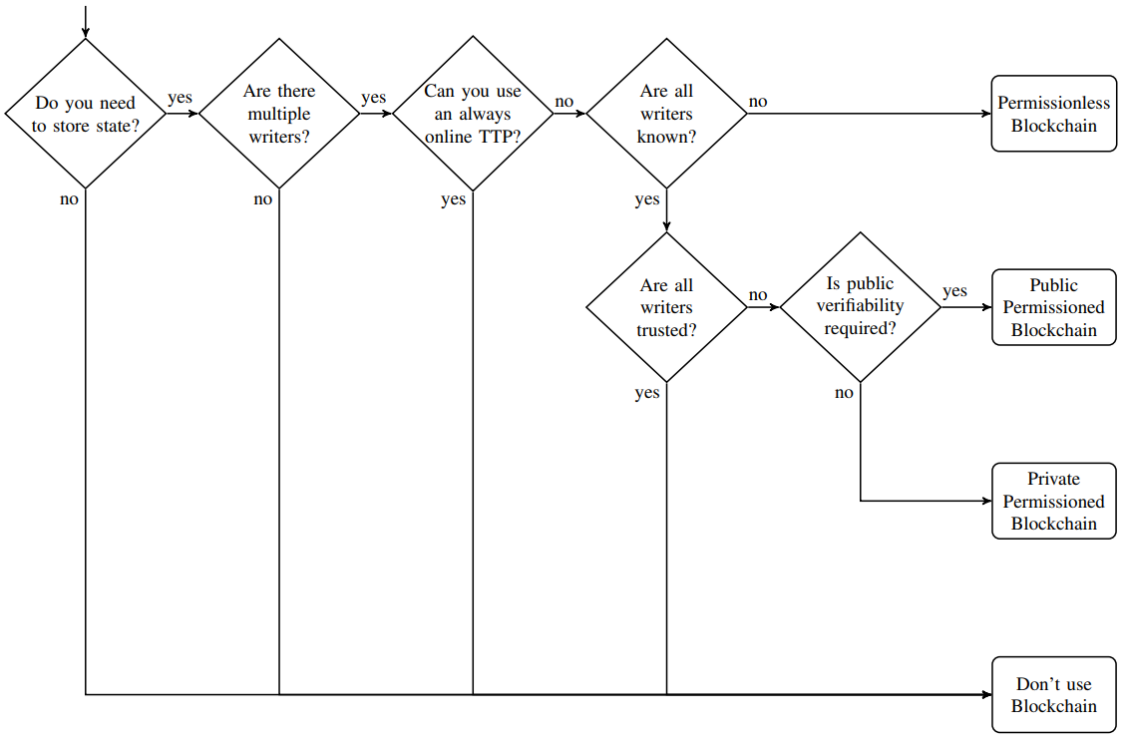


Figure 6.2: Decision-making process for determining the necessity of blockchain. The flow outlines key questions to assess if blockchain is required based on factors like the need for state storage, the presence of multiple writers, the availability of trusted third parties, and the requirement for public verifiability (from [65]).

6.3.1 A Blockchain... But Why?

While blockchain offers significant benefits in decentralization and security, its overuse in areas where simpler solutions could suffice has raised concerns about inefficiency, complexity, and unnecessary resource consumption. Critics argue that not all use cases warrant the high overhead and computational requirements that come with blockchain, especially in scenarios where traditional databases or federated systems could offer a more straightforward and effective solution. In decentralized systems like DeLoRaN, blockchain addresses several key challenges. As represented in Fig. 6.2, it provides a robust solution for storing network state and connected device data (e.g., EDs in LoRaWAN) while avoiding reliance on a SPoF. Distributed ledger enables multiple participants (like NCs) to independently verify and store transactions, ensuring data integrity and enabling parallel operations without central authority. Blockchain solves the issue of multiple writers by allowing NCs to interact with the network simultaneously, ensuring transaction validation through consensus. Additionally, in a decentralized environment, trust is a critical concern. DeLoRaN removes dependence on a trusted third party, instead relying on blockchain's consensus mechanism to ensure network functionality, even in the presence of malicious nodes (Byzantine nodes). Honest participants can outvote dishonest ones, enhancing resilience. Since all writers are known in DeLoRaN, blockchain ensures secure control distribution through consensus, mitigating risks from potentially compromised nodes. Therefore, a permissioned one is the best fit, providing decentralized control, security, and transparency without the overhead of a fully public system.

Alternatives

While blockchain emerges as a very suitable solution for decentralizing control in DeLoRaN, other alternatives exist but are less effective in multi-tenant environments where operators do not fully trust one another. One alternative is using centralized databases with redundancy. Although this approach can manage network data efficiently, it still depends on a central authority, creating a SPoF. This setup is vulnerable to attacks or internal breaches and requires complete trust in a single entity, which is impractical in a shared, multi-tenant network where no operator is willing to grant full control to one party.

Federated systems offer another alternative, where control is distributed among multiple entities that each manage their own part of the network. While this avoids centralization, federated systems still depend on trust between participants, as each entity must assume that the others will uphold agreements and behave fairly. Without a transparent, unified mechanism like blockchain, federated systems also lack immutability and verifiability. Similarly, a peer-to-peer network without blockchain might offer decentralized control, but it introduces risks, particularly in a multi-tenant scenario. Without blockchain's consensus mechanisms, peer-to-peer networks are prone to rogue nodes and data manipulation. In environments where no single operator controls the entire network, ensuring trust and preventing tampering becomes extremely challenging.

Blockchain, especially a permissioned blockchain, stands out in this context. It allows multiple known entities (like the NCs in DeLoRaN) to securely interact within a shared infrastructure, providing verifiability, trust, and transparency without the need for participants to fully trust each other. This model ensures that each entity can independently validate transactions and participate in decision-making, making it the optimal solution for decentralized, multi-tenant networks.

6.3.2 The Blockchain

Based on the previous motivations, we decided to rely on Hyperledger Fabric (HF)[1], a permissioned blockchain framework, as the foundation for the network's architecture. The choice of HF stems from its robust support for multi-organization environments, its flexibility in handling different types of access control, and its ability to efficiently manage a network where different entities need to collaborate without fully trusting one another. Unlike public blockchains, where anyone can join and participate in the consensus process, a permissioned blockchain restricts access to known, identifiable participants. This controlled environment is particularly suited to scenarios like DeLoRaN, where multiple network operators share infrastructure and need secure and verifiable transactions but do not want completely open access to the network. In a permissioned blockchain, all participants are vetted, and access is restricted based on roles, making it easier to enforce governance, privacy, and compliance with regulatory standards. This ensures that only authorized entities can manage network tasks and data while maintaining full accountability for their actions. HF was chosen because of different unique features that made it the best choice when deciding the kind of blockchain to rely on.

Permissioned Blockchain – In a permissioned blockchain, trust is still decentralized, but the entities involved are known and trusted to some extent, compared to anonymous participants in public blockchains. This is crucial in multi-tenant environments, where operators need to coordinate

[1]Hyperledger Fabric: https://www.hyperledger.org/projects/fabric

network activities and share resources but without giving up control or allowing full transparency to external entities. With HF, we achieve this balance, providing a system that ensures secure collaboration while safeguarding privacy and operational autonomy for each participant.

Modularity – HF provides a modular architecture that allows flexibility in its configuration. The network can be tailored to specific needs by customizing components such as consensus mechanisms, smart contracts (called chaincode), and identity management. This modularity is particularly advantageous for DeLoRaN, where different operators may have different requirements for security, performance, and compliance. Moreover the modularity allows HF to adapt in real time to changes, as its configuration, rules and even the chaicode running on it can be changed at runtime, without the need to restart or create a whole new blockchain when crucial changes are needed.

Role-Based Access Control – A key feature of permissioned blockchains is the ability to enforce fine-grained access control. In HF, access is governed by Membership Service Providers (MSP), which manage identities and determine who has the right to participate in the network. This is crucial in a shared network like DeLoRaN, where different operators need to control what parts of the network they can access or modify.

Privacy, Confidentiality and Integrity – HF provides enhanced privacy features by supporting private data collections and channels. In a multi-tenant scenario like DeLoRaN, where different operators may need to exchange data without fully exposing it to the entire network, these privacy features are critical. Channels allow for the creation of sub-networks where only certain participants can view or interact with specific transactions. Meanwhile, private data collections allow sensitive information to be shared among specific participants without being recorded on the main ledger. This is achieved by storing the actual data off-chain while cryptographic hashes of the data are placed on the blockchain to ensure integrity. This ensures that only authorized parties can access the sensitive information, while the rest of the network can verify that the transaction occurred without seeing the data itself. Data integrity is a core feature, ensuring that once data is written to the blockchain, it cannot be altered or tampered with. This is achieved through cryptographic hashing, where each block in the chain contains a hash of the previous block. Any modification to a block would result in a mismatch in the hash, immediately flagging the tampered data. Additionally, endorsing policies ensure that transactions are only accepted if they are validated by the required participants, adding another layer of security. These mechanisms protect the integrity of the entire system, making it resilient to tampering or unauthorized modifications.

Pluggable Consensus Mechanisms – HF supports a variety of consensus mechanisms, allowing network administrators to choose the consensus protocol that best fits their needs. In the case of DeLoRaN, where known participants operate in a semi-trusted environment, Crash Fault Tolerant (CFT) or Byzantine-Fault Tolerant (BFT) consensus mechanisms can be applied, depending on the level of fault tolerance required. This flexibility ensures that the network can balance performance with resilience against malicious actors.

Smart Contracts (Chaincode) – Chaincode allows for the automation of key processes such as authentication, device onboarding, and transaction validation. In a network like DeLoRaN, where numerous IoT devices and network interactions need to be validated and processed, chaincode can streamline these operations. Authentication of devices and validation of packets is performed by the chaincode itself, preventing malicious nodes from uploading invalid information and tamper with the data stored on the blockchain.

Scalability and Performance – HF's architecture is designed for high scalability, an essential feature for IoT networks like DeLoRaN that might need to handle thousands or millions of devices. Scalability is achieved through its modular architecture, which separates transaction processing into three distinct phases: endorsement, ordering, and validation. This separation allows for parallel processing, reducing bottlenecks typically seen in traditional blockchain networks. By allowing different nodes to handle endorsement and ordering independently, the network can manage a high volume of transactions without degradation in performance. This architecture ensures that as more devices or participants join the network, like in DeLoRaN, the system remains efficient, even under heavy load. Additionally, the use of channels and private data collections helps in compartmentalizing transactions, reducing the need for all participants to process all transactions, further enhancing scalability.

HF, with its permissioned blockchain structure, modularity, and focus on privacy, is an ideal choice for DeLoRaN. Its ability to manage known participants, enforce access control, and ensure secure, verifiable transactions in a multi-tenant environment makes it a robust and flexible solution for managing IoT networks. By leveraging HF, DeLoRaN achieves the decentralization necessary for resilient and secure network operations while maintaining the governance and privacy required for operator collaboration.

6.4 DeLoRaN Implementation

6.4.1 DeLoRaN Stack

The codebase for the DeLoRaN stack[2], which adheres to the LoRaWAN 1.1 standard [21], has been entirely developed using Rust[3], a language known for its focus on memory safety, concurrency, and performance. Unlike many alternative implementation of the LoRaWAN stack, it is fully backward compatibile with every standard version of LoRaWAN. This means that existing centralized LoRaWAN networks can transition to a distributed DeLoRaN setup without requiring any changes in the code of the EDs. From the point of view of the EDs, the communication protocol remains identical, allowing for a smooth transition to decentralization.

In the current implementation, all the key components needed for LoRaWAN operations are included. These encompass the EDs, a robust NC, and a basic AS. The NC is responsible for critical network operations, including ensuring packet integrity, handling encryption, managing nonce usage for both uplink and downlink communications and it is fully integrated with the backbone network. This makes the NC the first line of defense in securing the decentralized network.

The design also emphasizes modularity and extensibility. By adopting a modular architecture, the stack is future-proofed, enabling the integration of new features, protocols, and technologies. This adaptability ensures that DeLoRaN can evolve with the changing demands of IoT applications, making it a long-term, sustainable solution.

[2]DeLoRaN - https://github.com/DeLoRaN-Org/DeLoRaN
[3]Rust - https://www.rust-lang.org/

6.4.2 Blockchain - Chaincode and Collections

Due to the lack of a Rust SDK for HF, we decided to build a bridge. It sets up a local UDP server to listen for incoming messages and processes requests related to device management and data handling. The main tasks include handling the join procedure for devices, managing session generation, creating uplink messages, and interacting with device configurations and sessions stored on the blockchain. It uses chaincode to securely store and retrieve information like device configurations and packets, ensuring secure communication in the decentralized network. The server responds to incoming requests, processes them through blockchain transactions, and sends back the results to the requesting NC.

Chaincode – In HF, chaincode governs interactions with the stored data. In our implementation, two chaincodes play distinct but complementary roles:

- The **Device Chaincode** handles the management of connected EDs, covering the creation, updating, and deletion of device data. This ensures a consistent and up-to-date representation of the EDs across the network.
- The **Packet Chaincode** is focused on security, managing packet uploads and session updates after uplink communications. It includes security checks on encryption and integrity, reinforcing the network's defenses. Even if a NC is compromised, the integrity and security of the communication are preserved, preventing unauthorized actions on behalf of EDs.

Collections – Our implementation leverages both public and private storage through collections in HF. Each organization within the network also has two private collections, together with the shared public collection:

- **Security session data**: This private collection stores sensitive information, including cryptographic keys and nonce counters. These are essential for validating the integrity and authenticity of communications.
- **Historical packet data** (optional): This private collection maintains a detailed record of all packets sent and received. Storing this historical data enhances both security and network accountability, ensuring that sensitive information is managed securely without the need for additional distributed databases. This approach ensures the privacy and integrity of critical data while also enabling secure, private, and verifiable storage for all participants in the network.
- **Public Channel Collection**: it is used to store general device configurations, ownership details, and other non-sensitive information. This allows any NC to manage the device join process, even if the device is owned by another operator. Sensitive information, such as session keys and encryption details, is stored privately by the device's owner, ensuring that only they have access to it, maintaining a balance between flexibility in device management and the security of private data.

6.4.3 Performance and Privacy

HF's architecture is designed to support the high scalability demands of IoT networks [66], like DeLoRaN. By separating the transaction endorsement process from ordering, Fabric ensures efficient,

	Integrity	Availability	Confidentiality	Solved/Mitigated
Bit-flipping/MITM	✓	✓	**X**	**S**
Denial of Service	∼	✓	**X**	**M**
Sinkhole/Blackhole	**X**	✓	**X**	**M**
Roaming	✓	**X**	✓	**M**

Table 6.1: Security and resilience comparison for various attack vectors in DeLoRaN.

parallel processing of multiple transactions. This is essential in IoT applications where thousands or millions of devices need to communicate simultaneously. Fabric can handle this load without bottlenecks, allowing DeLoRaN to scale efficiently as the number of devices and operators in the network increases.

The use of private data collections also ensures that sensitive information remains confidential while still being verifiable through cryptographic hashes stored on the blockchain. This provides both privacy and security, making sure that critical data is only accessible to authorized parties without sacrificing the integrity of the overall network. Specifically, the collections are governed by strict privacy policies, ensuring that only authorized members of an organization can read or modify the data. Each transaction involving these collections must be endorsed by at least three other nodes from the same organization before being approved, ensuring a secure and verified process. Once endorsed, the data is stored across all the nodes of the organization, enhancing reliability and fault tolerance.

6.5 DeLoRaN Security Improvements

In addition to LoRaWAN security issued related to its centralization, there are also other vulnerabilities that DeLoRaN addresses. These vulnerabilities, presented in Tab. 6.1, affect different aspects of the network, including integrity, availability, and confidentiality:

- **Bit-flipping/MITM:** These types of attacks target the integrity of data by either flipping bits during transmission or intercepting communication to alter the content (Man-in-the-Middle);
- **Denial of Service (DoS):** DoS attacks aim to overwhelm the network by flooding it with requests, causing legitimate requests to be delayed or denied;
- **Sinkhole/Blackhole:** These attacks involve malicious nodes advertising themselves as attractive routes for data but then discarding or rerouting the data improperly;
- **Roaming:** Roaming refers to devices moving between different network domains. While the system supports such mobility, confidentiality could be at risk if the handover process is not securely managed;

6.5.1 Bit-flipping/Man-in-the-Middle (MITM) Attacks

Man-in-the-Middle attacks are a significant threat in networked systems, especially in large networks like LoRaWAN, where data is transmitted across multiple nodes. In a typical MITM attack, an

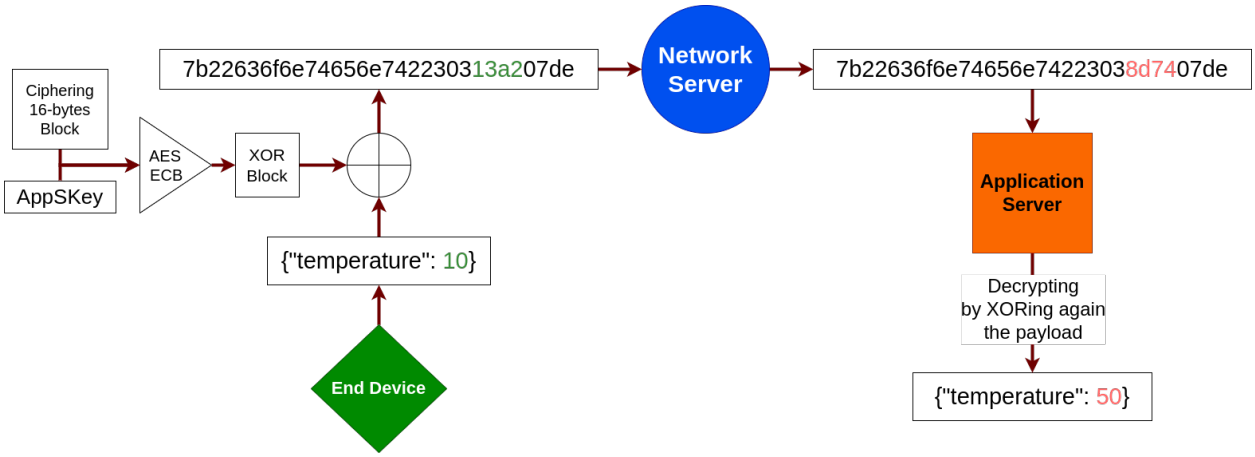


Figure 6.3: Flow of a MITM attack in a centralized LoRaWAN network. The compromised NS has full control of the payload and can alter the data before forwarding it to the AS. The AS is unable to detect the tampering, allowing the attacker to manipulate the communication without detection.

adversary intercepts communication between two parties, such as an ED and a NC, and may alter or inject malicious packets. This compromises both the integrity and confidentiality of the transmitted data. The attacker can eavesdrop on sensitive information or alter the contents of the message, leading to further security breaches.

In the context of traditional centralized LoRaWAN systems, MITM attacks are particularly dangerous due to the reliance on a central NS to authenticate and manage all communications. If the NS is compromised, the entire network can be exposed to widespread attacks. As shown in Fig. 6.3, a compromised NS can intercept and modify data packets between EDs and ASs, altering the contents of the messages without detection. This allows the attacker to manipulate the communication flow, inject malicious data, or eavesdrop on sensitive information, posing a significant threat to the network's security.

DeLoRaN provides a robust solution to this issue by leveraging blockchain technology to decentralize control and ensure message integrity. In DeLoRaN, each packet transmitted by an ED is cryptographically verified and stored on the blockchain before it reaches its final recipient, typically an AS. This decentralized storage ensures that the final recipient, such as the AS, does not need to rely solely on the NS or the integrity of the wireless transmission.

A key advantage of DeLoRaN's approach is that even if an attacker intercepts and modifies a message during transmission, the AS can retrieve the original, unaltered packet from the blockchain. The blockchain guarantees that the message stored is authentic because all integrity checks have been performed prior to the message being written onto the blockchain. As a result, the AS can trust the data retrieved from the blockchain, knowing it has passed all integrity tests and is protected from tampering. This mechanism prevents the attacker from injecting fraudulent data or altering the packet content, as any changes during transmission would be detected before the packet is added to the immutable blockchain ledger.

6.5.2 Denial of Service (DoS) Attacks

Denial-of-Service (DoS) attacks are a critical threat to IoT networks like LoRaWAN, where resource-constrained devices and centralized servers are vulnerable to being overwhelmed by excessive re-

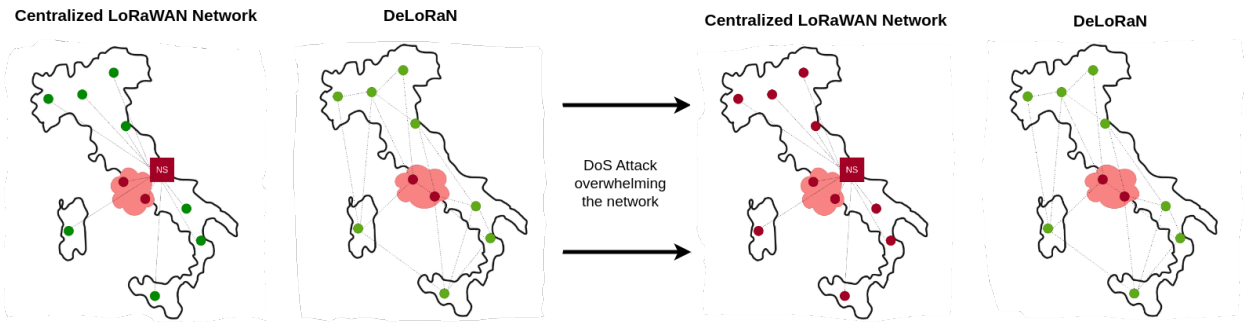


Figure 6.4: In a DoS attack, an attacker floods the NS with a large volume of requests, causing the system to become overwhelmed and unresponsive. Legitimate devices are unable to communicate effectively, leading to service disruptions and network downtime. In DeLoRaN instead, the downtime is only partial, as the network can still operate even if one or more NCs are targeted, causing only localized downtime.

quests. In a centralized LoRaWAN architecture, the NS acts as the central authority for all communication within the network. This makes it an attractive target for attackers, as a successful DoS attack can easily overwhelm the NS by flooding it with an excessive number of packets, rendering the entire network unusable. The consequences of such an attack can include prolonged downtime, delayed or failed communications, and reduced service availability for all connected devices.

In a typical DoS attack scenario, an attacker might continuously send a large volume of requests to the NS, consuming its processing power and bandwidth. Since the NS handles every message in a centralized system, once it is overloaded, legitimate devices are unable to communicate effectively, causing severe disruptions. The problem is compounded by the limited computational and bandwidth resources typical of LoRaWAN environments, making the network highly susceptible to such attacks.

DeLoRaN offers a significant improvement over centralized systems by distributing control across multiple NCs using blockchain technology. In a decentralized architecture, there is no single point of failure, and managing network communications is distributed across several NS. This makes it considerably more difficult for an attacker to launch a successful DoS attack, as overloading one NC will not cripple the entire network. The decentralized nature of DeLoRaN ensures that even if one or more NCs are targeted in a DoS attack, other NCs can continue to process network traffic, ensuring that communications can still proceed.

While decentralization alone does not completely eliminate the threat of DoS attacks, it greatly enhances the network's resilience by ensuring that the system does not depend on a single server. By spreading the load across multiple NS, DeLoRaN can continue to function even under attack, reducing the overall impact on the network's availability and minimizing the risk of total service disruption.

6.5.3 Sinkhole/Blackhole Attacks

Sinkhole and Blackhole attacks represent a serious threat [67], since routing of data packets plays a critical role in network functionality. In these attacks, a malicious node such as a compromised Gateway (GW), falsely advertises itself as the optimal route for packet transmission, attracting network traffic. Once the traffic is routed through this malicious node, it can either drop all the packets (Blackhole) or selectively forward some while discarding others (Sinkhole). The result

is a significant disruption in communication, as valid data from EDs never reaches its intended destination, causing failures in application layer processing and data loss.

In a traditional centralized LoRaWAN architecture, this type of attack can be particularly devastating because once the malicious node becomes the designated route for data, the NS has no easy way to verify whether the transmitted data has been successfully forwarded to the correct destination. The NS's central role in communication makes it vulnerable to such attacks, as any interference in routing between the ED and the NS can cause network-wide disruptions.

In a decentralized system, traffic is managed by several nodes rather than a single point, making it more difficult for a malicious node to disrupt the network. If one node starts behaving suspiciously by dropping or misrouting packets, the remaining NCs can detect this anomaly and exclude the node from the routing process. This process significantly reduces the potential impact of Sinkhole and Blackhole attacks because no single node has full control over the network's traffic.

DeLoRaN introduces elements of pseudorandomized routing and BFT consensus to complicate the attacker's ability to predict traffic paths. By not relying on static routes, the system makes it difficult for attackers to target specific data streams. This unpredictability, combined with continuous monitoring of node behavior, creates a dynamic and resilient environment where the impact of any single malicious node is minimized.

Pseudorandomization and Consensus

To solve the problem of static routing/hanlding of the communications, DeLoRaN must firstly solve the two main centralized operations: deduplication of the packets, thus uplink handling, and the join procedure. The solution is to use a pseudorandomized routing mechanism, combined with a consensus algorithm, to ensure that the network can operate securely and efficiently even in the presence of malicious nodes.

Join Consensus – The process, represented in Fig.6.5, begins with a join request being sent from the ED to multiple NCs, each of which performs security and integrity checks. Once the checks are completed, the join request is transmitted to the blockchain network, where it undergoes transaction processing. A deduplication window of 500ms is provided to allow multiple NCs to process the same request, storing it in a temporary transaction. Following this, the NC set that received the join request is merged and retrieved. A downlink NC is selected based on the MIC using a deterministic, yet unpredictable, algorithm, ensuring load distribution and making it virtually impossible for the attacker to know in advance the selected NC. The session context is derived by the blockchain chaincode, which securely stores the join request and join accept message. This process allows the ED to join the network securely, with blockchain providing a robust mechanism for session context derivation and ensuring the integrity of communications.

Uplink Consensus – The process, represented in Fig.6.6, begins with each NC performing security checks and verifying the integrity of the received message. Once this is complete, the consensus mechanism is initiated. The first phase involves the exchange of proofs of reception among NCs to confirm which controllers successfully received the message. In the second phase, the NCs exchange reception sets, which include information about the message reception status. This is followed by filtering out any NCs that did not receive more than 66% of votes. Finally, a downlink NC is selected based on the packet's MIC, using the same logic as the join consesus, ensuring that the network can complete the communication process in a decentralized, secure manner. This

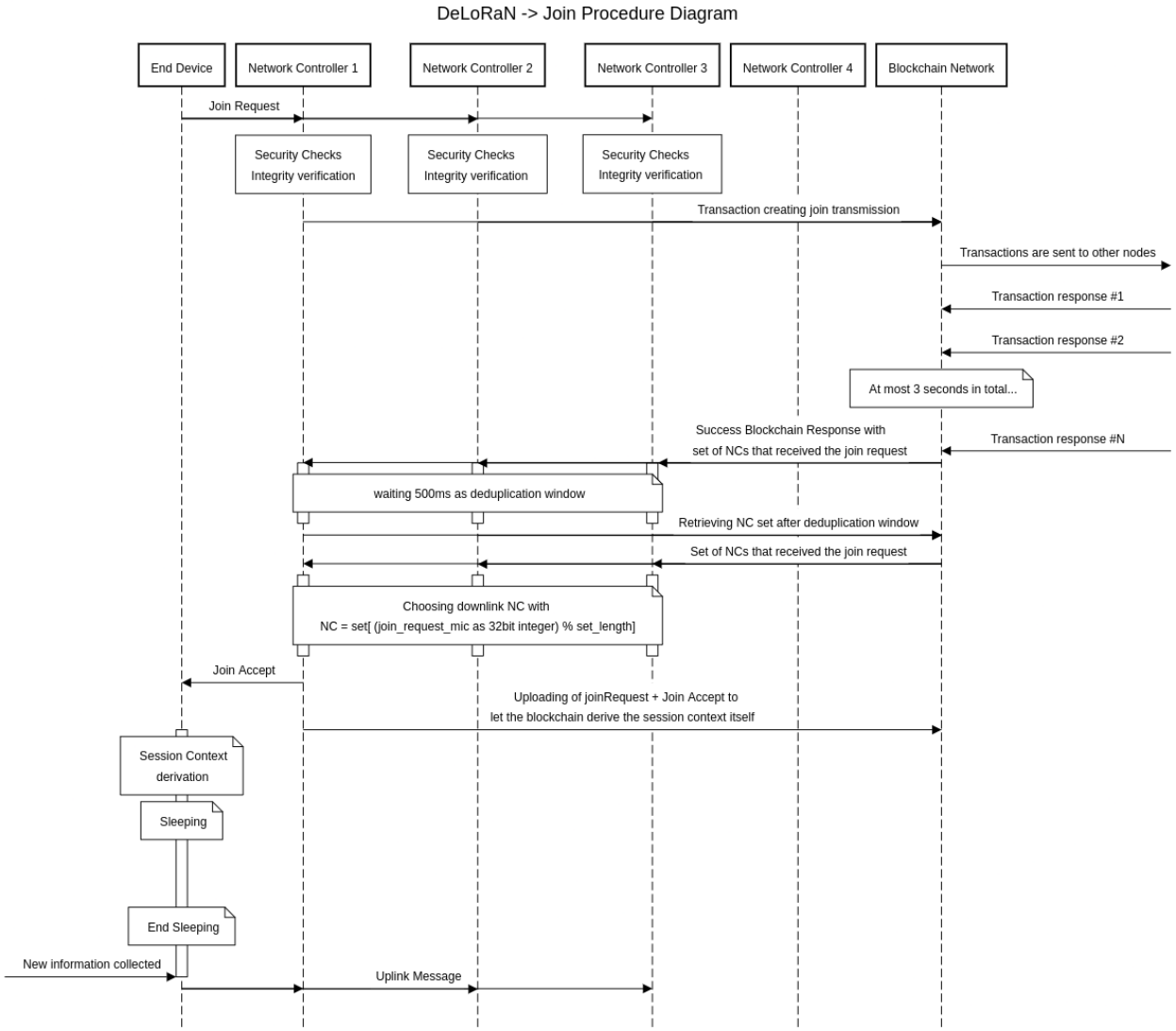


Figure 6.5: The join consensus process begins when a new device attempts to join the network. During this process, the blockchain serves as an intermediary platform to establish a set of listening NCs. This set of NCs will be later utilized to facilitate the uplink consensus procedure, ensuring efficient and coordinated network operations.

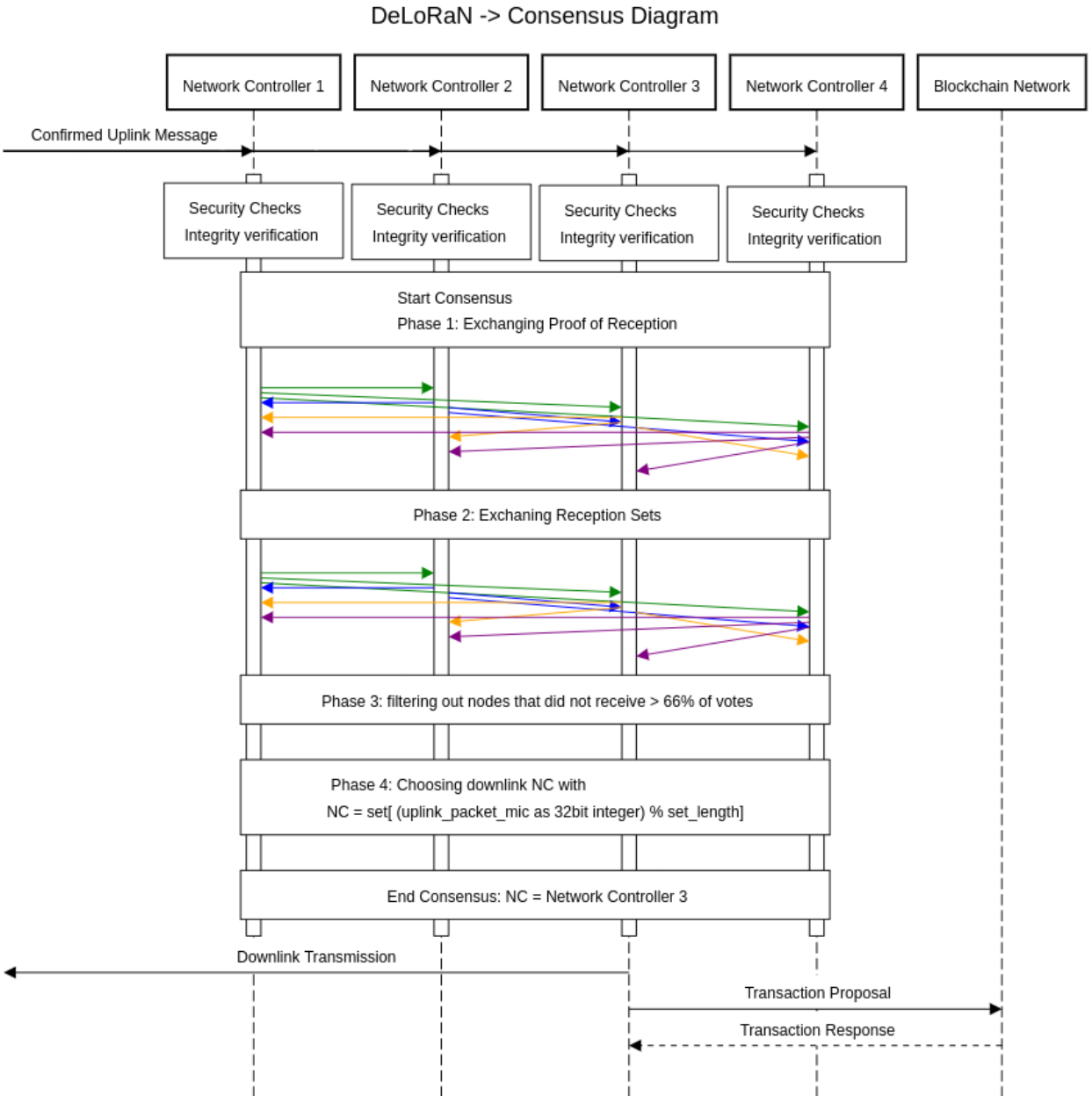


Figure 6.6: The process of uplink consensus begins with each NC verifying the message's integrity and initiating consensus. NCs exchange proofs of reception and reception sets, filtering out those without 66% votes. The Message Integrity Code (MIC) determines the downlink NC, ensuring secure, decentralized message validation and eliminating single SPoFs.

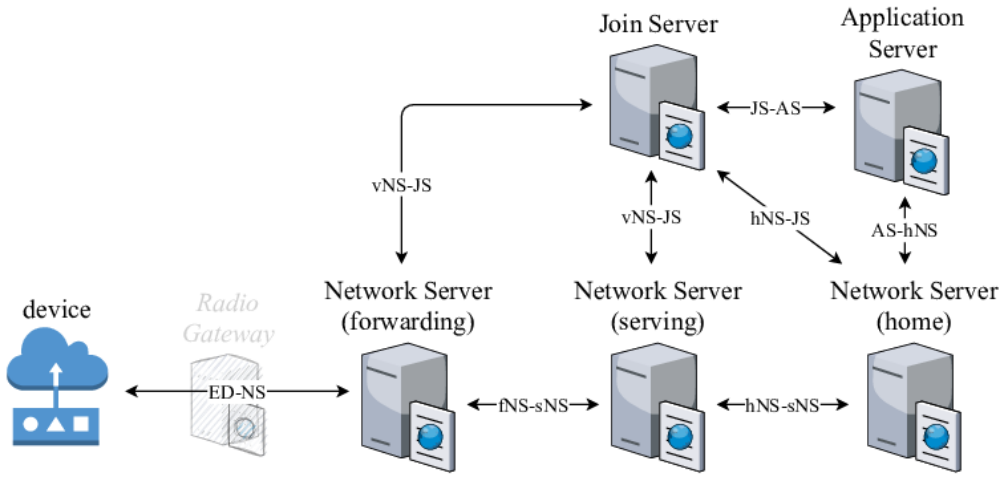


Figure 6.7: In LoRaWAN Roaming, a device moves between different network domains, requiring secure handover to maintain confidentiality. It uses different NSs, Forwading, Serving or Home, depending on the context (from [68]).

process eliminates any SPoF by distributing message validation across multiple NCs, which enhances network reliability and security.

The combination of these techniques ensures that DeLoRaN is highly resilient to Sinkhole and Blackhole attacks, providing greater security and operational stability compared to traditional centralized systems.

6.5.4 Roaming

LoRaWAN roaming, shown in Fig. 6.7 introduces two primary types of roaming mechanisms: handover and passive roaming. Each comes with its own set of challenges and benefits. In handover roaming, devices switch their connection completely to the visited network, entrusting the visited NS to manage all network tasks. This method, though seamless in terms of switching, depends heavily on trust, as the home network must relinquish control to the serving NS.

In contrast, passive roaming allows devices to remain connected to their home network, even when using gateways from a visited network. This is further divided into stateful and stateless approaches. In the stateful method, the home network takes charge of session management, ensuring roaming across different networks, but this comes with significant overhead in terms of session handling. Stateless passive roaming simplifies this by forwarding packets based on identifiers, but leads to indiscriminate packet forwarding to the home network, based solely on the NetID identifier, present in the uplink header.

However, the introduction of roaming also brings various security concerns. Roaming demands a high degree of trust between operators. In an inter-operator setting, private session keys are shared, creating vulnerabilities. The visited network can not always validate the transmissions it forwards, the home network can not dispute them, leading to possible waste of money for both the networks. Moreover, attacks like MITM, carried on by the serving LoRaWAN network or intermediate nodes, pose serious risks. Additionally, the intermediate nodes cant still carry on blackhole/sinkhole attacks. Another critical issue is the reliance on centralized authorities, such as roaming hubs, which may become SPoFs or require a substantial level of trust among participating operators.

Blockchain provides a robust solution to many of these issues. By decentralizing the control and using a shared ledger, DeLoRaN ensures that trust between operators is no longer a requirement. Ownership of devices is validated through the blockchain, making it impossible for an operator to deny a device's presence or claim. This also introduces the concept of "hard proof of reception," where a NC can validate that a packet was indeed received. Blockchain's immutable nature ensures that even in the presence of malicious entities, data integrity remains intact, as all participants can verify the authenticity of the information. Furthermore, chaincode automates the roaming process, enabling seamless and transparent coordination between operators without requiring direct trust.

In summary, the use of blockchain in DeLoRaN significantly enhances the security of LoRaWAN roaming by decentralizing trust, ensuring data integrity, and automating critical processes through smart contracts, all of which are particularly valuable in multi-operator environments.

Chapter 7

DeLoRaN - Experimental Evaluation

7.1 Introduction

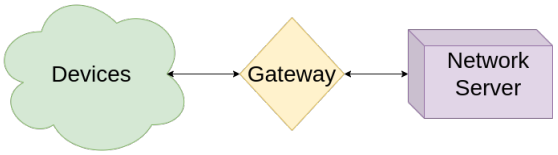
The experiments described in this chapter aim to comprehensively evaluate the performance, security, and scalability of DeLoRaN under various conditions and scenarios. These tests were designed to assess how effectively DeLoRaN decentralizes control and addresses the inherent limitations of traditional centralized LoRaWAN networks.

One of the core experiments focused on comparative performance evaluations between DeLoRaN, a traditional centralized LoRaWAN system and a blockchain-integrated LoRaWAN network named HyperLoRa [69]. These tests provided a detailed comparison, highlighting improvements in areas such as latency reduction, transaction throughput, and enhanced security, all contributing to DeLoRaN's advantages in decentralized network management.

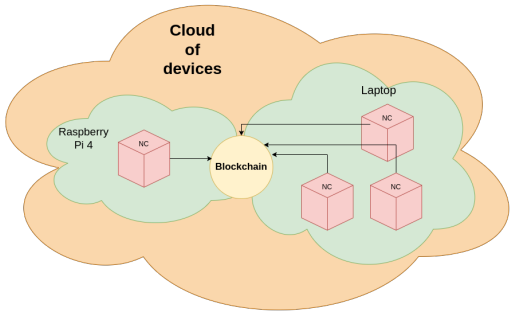
Additionally, we carried out experiments focused on scalability, analyzing how the system performs with an increasing number of End Devicess (EDs) and Network Controllers (NCs). By incrementally scaling the number of EDs in the network, we evaluated the system's ability to handle large-scale deployments, typical of smart city applications or industrial Internet of Things (IoT) environments. The tests confirmed DeLoRaN's ability to efficiently manage high device loads without bottlenecks or performance degradation, highlighting the advantages of blockchain-based control distribution.

In parallel, AI-based interferences/jamming detection was explored as a starting point for a smart and distributed Adaptive Data Rate (ADR) implementation, where statistical and machine learning models were employed to detect and mitigate interferences/jamming attacks on the network. These tests were conducted using ns-3[1] simulations, offering insights into how DeLoRaN can enhance resilience against radio frequency interference.

The results from these experiments confirm that DeLoRaN is not only scalable and secure but also resilient to a variety of network threats and capable of handling the operational demands of modern IoT environments.



(a) Centralized LoRaWAN Setup. It consists of a cloud of EDs, a Gateway (GW) bridge, and a central Network Server (NS) that manages the network's data and devices.



(b) DeLoRaN Setup. It consists of a distributed network of four NCs, each connected through the blockchain network.

Figure 7.1: Centralized LoRaWAN vs DeLoRaN Setup. The first round of tests compared the performance of a traditional centralized LoRaWAN network with a decentralized DeLoRaN network.

7.2 Centralized vs Decentralized LoRaWAN

The first set of experiments focused on comparing the performance of DeLoRaN with traditional centralized LoRaWAN networks, as shown in Fig. 7.1, with both scenarios tested on a Raspberry Pi 4[2]. The centralized solution is based on the most used, open-source implementation of a LoRaWAN stack by ChirpStack[3], which includes a cloud of EDs, a GW bridge, and a central NS that manages the network's data and devices. In this setup, presented in Fig. 7.1a, all devices send their data to the GW, which forwards it to the NS, which handles the uplinks and send the corresponding downlink back to the EDs. Everything except the devices was running on top of the Raspberry.

In contrast, the decentralized solution in DeLoRaN leverages a distributed network of four NCs, each connected through the blockchain network. In this architecture, the processing and management of data are distributed among multiple controllers. Only one of the NCs was running on top of the Raspberry, while the other NCs and the EDs cloud were simulated on a different machine. The setup is illustrated in Figure 7.1b. It is worth noticing that the single NC hosted on the raspberry still receives and process all the data from the EDs cloud, but it is not always responsible for processing all the transfer through the blockchain or the handling of downlink messages.

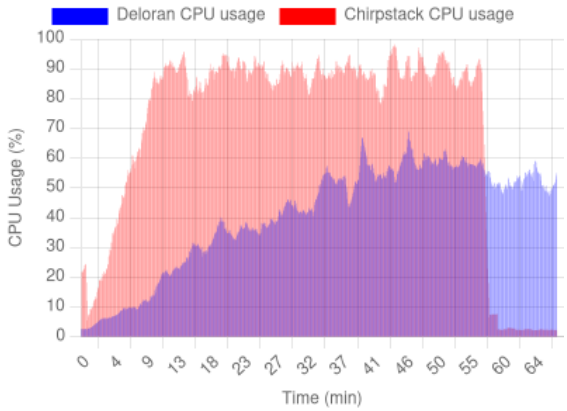
Both solutions were tested under varying load conditions, scaling the number of devices (500, 1000 and 2000), to evaluate how each system performs under increased demand.

7.2.1 Performance Indicators

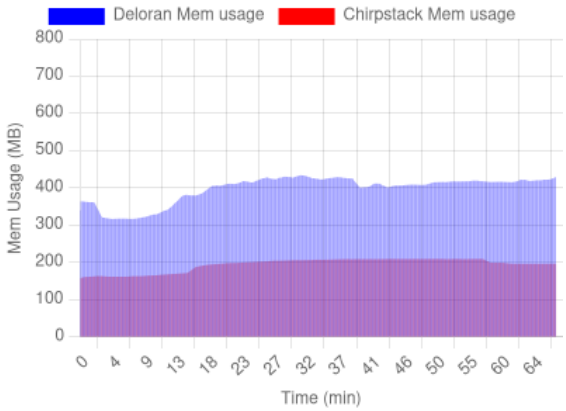
When evaluating the performance of both centralized and decentralized LoRaWAN architectures, two critical indicators emerge: delays and resource utilization.

Resource utilization – The focus is on measuring how much CPU, RAM, and network bandwidth is consumed by each architecture during the experiments. In the centralized Chirpstack architecture, the single network server handles all data, simplifying the overall network structure and communication patterns. Nodes of the network are simulated using Docker[4] containers, so

[1]NS-3 - https://www.nsnam.org/
[2]Raspberry Pi 4 - https://www.raspberrypi.com/products/raspberry-pi-4-model-b/
[3]Chirpstack - https://github.com/chirpstack
[4]Docker - https://www.docker.com/



(a) CPU Usage DeLoRaN vs Centralized.



(b) RAM Usage DeLoRaN vs Centralized.

Figure 7.2: Example of RAM and RAM Usage DeLoRaN vs Centralized. It shows the CPU and RAM usage of the Raspberry Pi 4 when running the centralized setup and the DeLoRaN setup with 2000 devices.

Devices	Chirpstack %CPU Usage	DeLoRaN %CPU Usage	CPU Comparison	Chirpstack RAM Usage (MB)	DeLoRaN RAM Usage (MB)	RAM Comparison
500	24.79	11.76	↓ 54.16%	129.07	305.36	↑ 136.58%
1000	41.38	21.65	↓ 48.88%	154.12	328.38	↑ 113.22%
2000	68.76	35.06	↓ 49.01%	194.94	409.38	↑ 110%

Table 7.1: Comparison of CPU and RAM usage between Chirpstack and DeLoRaN for various device loads, with arrows indicating better (down) or worse (up) performance for DeLoRaN

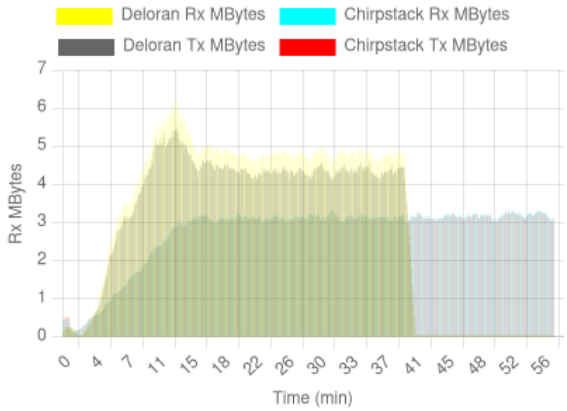
Docker tools were used to gather these metrics. On the other hand, in the decentralized DeLoRaN setup, where network controllers were simulated using LXD[5] containers, Linux tools were used to measure CPU and RAM utilization. Understanding how the resources scale when device numbers increase is critical for assessing the viability of each architecture in real-world IoT scenarios.

Delays – They play a significant role in determining the responsiveness and reliability of the network. In LoRaWAN, strict time constraints govern when devices can receive downlink messages, meaning that the delays between sending an uplink message and receiving the corresponding downlink must fall within predefined time windows. The traffic model used is a confirmed uplink with a random delay between 60 and 180 seconds between each transmission, sending around 30 bytes of data. With this pattern, a device almost reaches its daily quota (0.1%) of transmission time. These delays were measured during the experiment, both to compare the efficiency of the centralized and decentralized architectures and to assess whether the decentralized implementation can comply with the time-critical nature of LoRaWAN, ensuring that all responses fit within the allocated time window for receiving downlink messages.

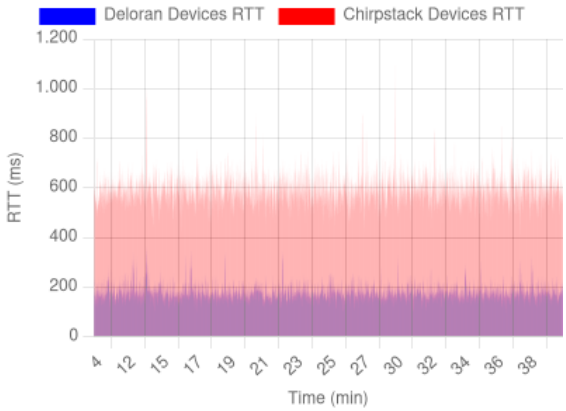
7.2.2 Results

CPU and RAM Usage Analysis – The results for both CPU and RAM usage across different device loads, summarized in Tab. 7.1, highlight the trade-offs between centralized and decentralized architectures in handling IoT networks.

[5]LXD - https://canonical.com/lxd



(a) Network Resources Utilization. It shows the network bandwidth usage in terms of MB/s sent and received by the node. It highlight how a decentralized network must take into account an increased network complexity, thus additional network overhead.



(b) RTT DeLoRaN vs Centralized. It shows the round-trip time for the network, highlighting how greately reduced is the latency in a decentralized network, thanks to the NC being much closer to the EDs instead of running in a centalized cloud.

Figure 7.3: Network Resources Utilization and RTTs of DeLoRaN vs Centralized

For CPU usage, we see a significant reduction in load when switching from Chirpstack to DeLoRaN. With 500 devices, Chirpstack's average CPU usage was 24.79%, whereas DeLoRaN reduced this load to 11.76%, reflecting a reduction of 54.16%. Similarly, with 1000 devices, Chirpstack showed 41.38% CPU usage, while DeLoRaN used only 21.65%, resulting in a 48.88% reduction. This trend continued with 2000 devices, shown in Fig. 7.2a, where Chirpstack's CPU usage reached 68.76%, compared to DeLoRaN's 35.06%, yielding a 49.01% reduction. The decentralized structure of DeLoRaN helps distribute computational tasks more effectively, reducing the CPU burden on any single network controller, making it a highly scalable solution for large IoT networks.

On the other hand, RAM follows a different trend. Decentralization tends to increase memory consumption due to the additional data processing due to the blockchain. With 500 devices, Chirpstack used 129.07MB of RAM, whereas DeLoRaN consumed 305.36MB, reflecting an increase of 136.58%. For 1000 devices, Chirpstack used 154.12MB, while DeLoRaN required 328.38MB, showing a 113.22% increase in RAM usage. With 2000 devices, represented in Fig. 7.2b, Chirpstack's RAM usage was 194.94MB, while DeLoRaN reached 409.38MB, marking an increase of 110%. While decentralization comes with increased memory demands, a 50% reduction in CPU usage presents a highly favorable tradeoff for increased RAM consumption, especially considering that even low-end devices today typically come with at least 1GB of RAM, making the additional memory demand manageable in exchange for significantly improved processing efficiency.

Network Resources Usage Analysis – The network throughput for both transmission (Tx) and reception (Rx) in DeLoRaN showed an increase compared to the centralized Chirpstack architecture. Specifically, with 500 devices, DeLoRaN experienced a 65.56% increase in Tx throughput over Chirpstack. As the number of devices grew to 1000, this increase dropped to 51.39%, and further to 46.62% with 2000 devices.

This increase in network throughput is a result of the additional complexity introduced by DeLoRaN's decentralized architecture, particularly its blockchain mesh network. The need for continuous communication between multiple NCs and blockchain nodes to maintain synchronization

Number of Devices	Chirpstack Tx Throughput (Mbps)	DeLoRaN Tx Throughput (Mbps)	Tx Throughput Comparison (%)
500	1.00	1.66	↑ 65.56%
1000	2.10	3.18	↑ 51.39%
2000	3.50	5.13	↑ 46.62%

Table 7.2: Comparison of Tx throughput between Chirpstack and DeLoRaN for various device loads, with percentage increase in throughput for DeLoRaN

Number of Devices	Chirpstack Processing Time (ms)	DeLoRaN Processing Time (ms)	Processing Time Comparison (%)
500	617.39	181.72	↓ 70.57%
1000	621.20	181.35	↓ 70.81%
2000	598.52	178.58	↓ 70.16%

Table 7.3: Comparison of Processing Time between Chirpstack and DeLoRaN for various device loads, with percentage decrease in processing time for DeLoRaN

and consensus leads to higher network consumption. This added overhead is inherent in a distributed setup, where data and control tasks are spread across various nodes to enhance security and fault tolerance. Despite this, the trend indicates that the relative increase in throughput diminishes as the number of devices scales up, suggesting that the decentralized architecture may scale efficiently. As more devices join the network, the percentage increase in network consumption becomes smaller, pointing toward more stable network demands in larger deployments. Importantly, the observed increase in throughput remains within an acceptable range, particularly considering the trade-offs that decentralization offers in terms of resilience and scalability.

Thus, while decentralization adds overhead in terms of network traffic, the scalability demonstrated by DeLoRaN highlights its potential for large-scale IoT deployments, where such overhead becomes more manageable as the system grows.

Response times – As shown in Tab. 7.3, the response times for Chirpstack remain consistent across all three scenarios, 500, 1000, and 2000 (represented in Fig. 7.3b) devices, the response times for Chirpstack remain consistent at around 620ms, while DeLoRaN consistently maintains a response time of approximately 180ms. This represents an overall improvement of around 70% in processing times with DeLoRaN. This significant difference in performance can be attributed to the decentralized nature of DeLoRaN, which efficiently distributes the workload. However, the most impactful factor is the elimination of the extra step involving the gateway present in Chirpstack's centralized architecture. By bringing the network control closer to the edge, DeLoRaN demonstrates the clear advantages of edge computing, particularly in massive IoT scenarios where reducing latency is critical for system performance and scalability.

7.3 Scalability of DeLoRaN

In this next series of tests, the focus shifts to evaluating the scalability of the decentralized DeLoRaN architecture. The tests aim to analyze how the network performs under increasing load

and how various key parameters, such as the number of devices and NCs, affect system scalability. By progressively increasing the number of network controllers and devices, this round of testing investigates the performance bottlenecks that may arise in large-scale deployments. The key metrics being evaluated are network delays and resource utilization, which will allow us to assess the system's ability to scale efficiently while maintaining low response times and handling significant traffic loads. Understanding how the system behaves under different configurations is critical to evaluating DeLoRaN's potential for scaling in real-world IoT deployments.

These tests have been performed using Colosseum[6], housed at Northeastern University, which is the world's most powerful hardware-in-the-loop network emulator. Colosseum provides a platform for testing and evaluating wireless networks by emulating real-world scenarios with extreme accuracy. With 128 independent nodes, each equipped with a Dell server, NVIDIA GPUs, and software-defined radios, Colosseum can simulate environments ranging from crowded urban areas to remote open fields. A key component of Colosseum is its Massive Channel Emulator (MCHEM), which can emulate up to 256 x 256 independent wireless RF channels, allowing researchers to observe how wireless signals interact in various conditions. This powerful emulation capabilities makes Colosseum ideal for large-scale DeLoRaN experiments, allowing our network to scale up without limitations to explore the performance, throughput, and delay under heavy loads, allowing us to use a single node as NC. In this way the NCs are isolated from each other and the EDs, allowing us to focus on the scalability of the network.

7.3.1 Description of the experiments

In these experiments, we varied three key parameters to assess the system's performance: the number of NCs, the number of emulated EDs and the number of orderers (nodes responsible for transaction ordering in the blockchain). The NCs were deployed in LXC containers, ranging from 2 to 8 in number, and each NC handled from 200 up to 1600 devices. Similarly, the orderers were also deployed in LXC containers, with their count varying from 4 to 16, to evaluate their impact on blockchain transaction processing and overall system latency.

The experiments revealed that scaling the number of orderers did not significantly affect performance. Across all tested configurations, the number of orderers had minimal impact on blockchain transaction times or overall system latency. As a result, for the scale of our simulation, the number of orderers proved to be a non-critical factor. Consequently, this parameter will not be analyzed further in the discussion, as it did not introduce any observable performance differences in our tests. All the tests shown in this section were performed using 8 orderers.

The primary performance metrics of interest in this round of tests included EDs response times, blockchain transaction times, and network resource utilization. These metrics provided insight into how DeLoRaN's decentralized architecture handles increasing loads, both in terms of the number of devices and the number of network components managing those devices.

By testing these configurations, we sought to answer critical questions about the scalability of DeLoRaN, particularly how well it can distribute the load across multiple controllers and blockchain nodes without suffering performance degradation. The results of these tests offer valuable insights into the system's robustness and its capacity to efficiently manage large-scale IoT deployments.

[6]Colosseum - https://www.northeastern.edu/colosseum/

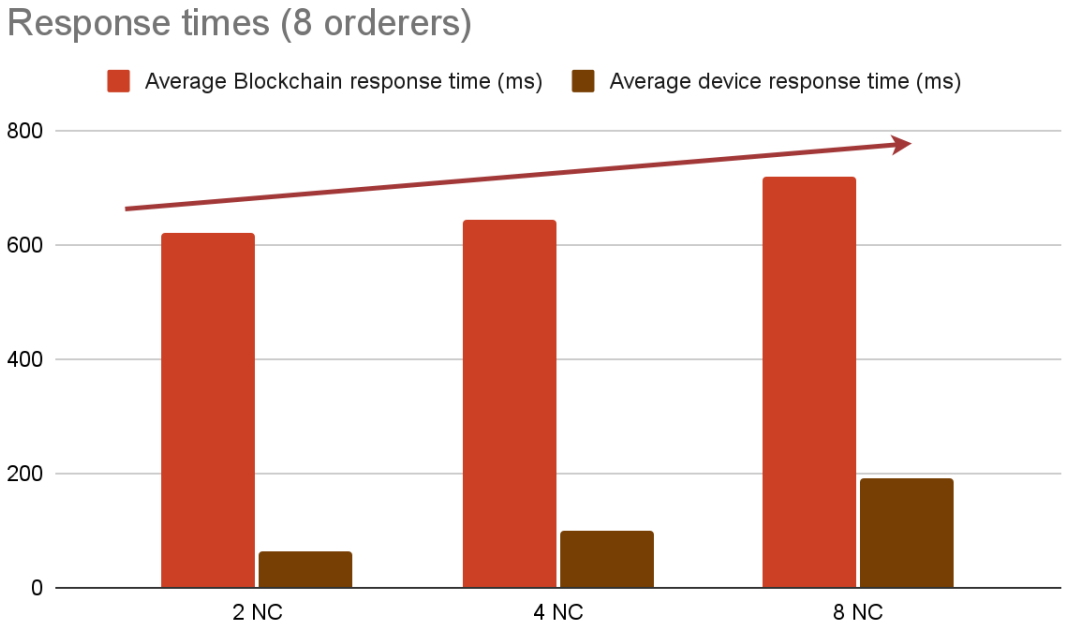


Figure 7.4: DeLoRaN delays with increasing NCs number. It shows the response times of the network controllers as the number of controllers increases exponentially. The response times increased slightly across all configurations, indicating that the system can efficiently distribute the load across multiple controllers without significant performance degradation.

7.3.2 Results

Scaling up Network Controllers – The results from scaling the number of NCs provide insights into the system's response time behavior. As shown in Fig. 7.4, increasing exponentially the number of NCs from 2 to 8 results in a slight gradual increase in the average blockchain response time. For example, with 2 NCs, the average response time is around 600 ms, while with 8 NCs, it increases to approximately 800 ms. This increase can be attributed to the additional communication overhead introduced by the larger number of nodes that need to reach consensus. Interestingly, the device response time remains relatively stable across all configurations, suggesting that the added complexity in the blockchain network does not significantly affect the network performance from the EDs point of view. This demonstrates that while the blockchain network experiences higher response times due to the increased coordination between NCs, the EDs continue to operate efficiently. These results highlight that, at least for the scale of our experiments, increasing the number of NCs introduces overhead but does not degrade edge performance.

Network Resource Utilization – As we scaled the number of NCs from 2 to 8, the network utilization, measured in both transmitted and received data, increased significantly. With only 2 NCs, the transmitted data was around 100 MB, and received data was slightly higher. As the number of NCs doubled to 4, we observed an almost linear increase in both transmitted and received data, reaching around 200 MB transmitted and 300 MB received. Finally, at 8 NCs, the network traffic continued to rise, with over 300 MB of transmitted data and nearly 400 MB of received data.

In analyzing the network resource utilization, we observe a clear pattern of increasing resource consumption as the number of NCs scales up. As the network expands from 2 NCs to 8 NCs, both transmitted and received data volumes rise steadily. However, this increase in network usage

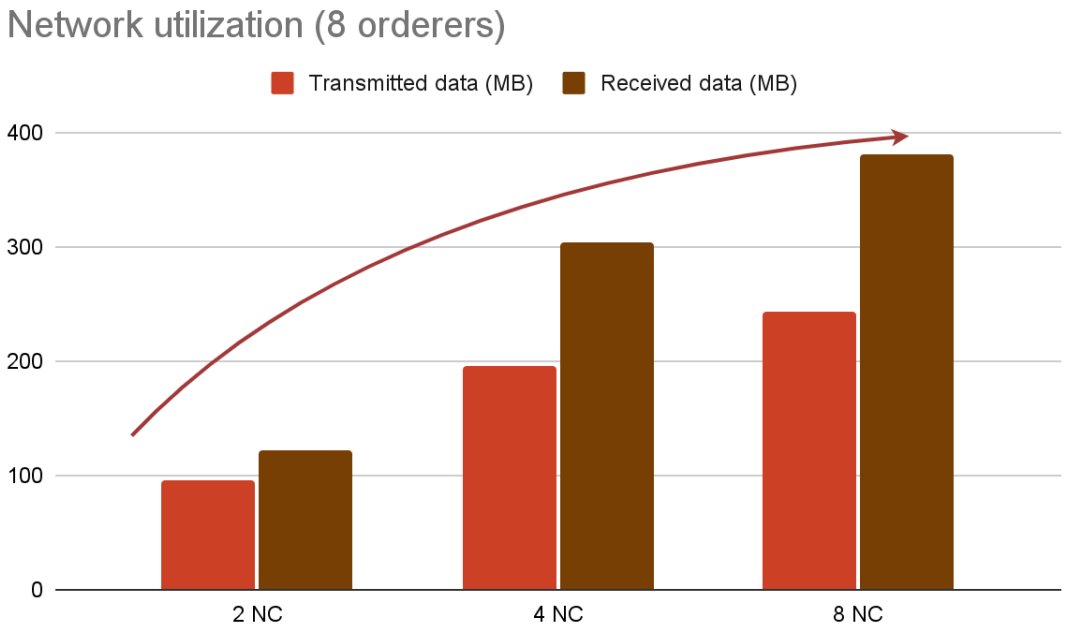


Figure 7.5: DeLoRaN Network resource utilization across scaling NCs. It shows the network bandwidth usage in terms of MB/s sent and received by the node. The network utilization increases linearly as the number of controllers grows exponentially, indicating that the system can efficiently manage an increasing backbone network.

remains linear and does not exhibit the exponential growth that is often expected in fully connected mesh networks, where network usage typically scales exponentially with the number of nodes. The initial increase in network resource utilization is linear, and the rate of increase diminishes as the network scales further. This behavior suggests an efficient management of transmissions, likely due to the effective implementation of a gossiping protocol. As a result, the system shows promising scalability in terms of network resource utilization, indicating that the decentralized architecture can handle larger networks efficiently, without overwhelming the communication channels.

Scaling up Devices – The final plot illustrates an intriguing and somewhat counterintuitive outcome regarding blockchain response times as the number of devices increases. As shown in the graph, as the number of devices rises from 200 to 1600, the blockchain response time decreases steadily. This phenomenon occurs because a higher number of devices leads to a greater number of transactions, which in turn fills the blockchain blocks faster. With a faster rate of block completion, the transactions are confirmed more quickly, improving the overall response time of the blockchain network.

It also highlights an important aspect of DeLoRaN's architecture, that is the disjunction between blockchain response time and device response time. The NCs handles everything related to the ED before interacting with the blockchain, which is used to update the ED session on the blockchain and broadcast the update to all the others NCs. This separation allows for a flexible and efficient handling of blockchain processes without negatively impacting device communication. In practice, this means that even with a high block timeout, used to minimize the number of blocks built and, therefore, reduce overall network load, the device response time remains low. The high block timeout slows down the blockchain's network utilization, ensuring fewer blocks are built and reducing the

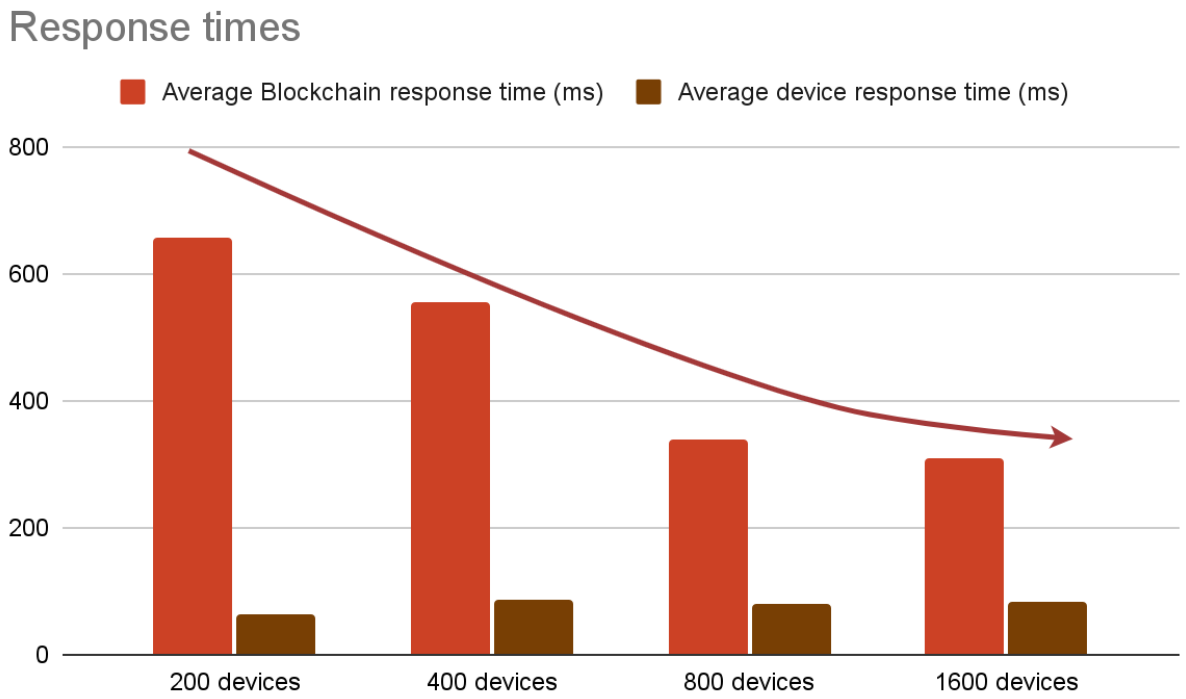


Figure 7.6: DeLoRaN delays with increasing devices number. It shows the response times of the network controllers as the number of devices increases exponentially. The response times remain relatively stable across all configurations, indicating that the system can efficiently manage an increasing number of devices without significant performance degradation.

overall bandwidth required for block propagation. This results in a significant reduction in network utilization without compromising the speed of device communications.

Furthermore, these tests have underscored the critical role of blockchain parameters such as block size and block timeout. By tuning these parameters, we were able to achieve increased efficiency in the blockchain network under higher transaction loads. The adjustments, specifically increasing the block size and timeout, led to a substantial boost in network efficiency while significantly reducing the overall network load. As a result, the decentralized system demonstrated its ability to not only handle but thrive under high-demand conditions, proving the importance of optimizing blockchain parameters in large-scale IoT deployments.

In conclusion, the experiments demonstrate that DeLoRaN scales efficiently across a range of network configurations and device loads, proving its resilience and adaptability in large-scale IoT deployments. The system handles an increasing number of devices without overwhelming network resources, maintaining consistent device response times while effectively reducing blockchain response times as the load increases. This counterintuitive result highlights the strengths of DeLoRaN's decentralized architecture, where the blockchain fills blocks more rapidly under heavier load, improving overall throughput without negatively impacting device communications. Furthermore, the clear separation between blockchain and device response times allows for flexibility in configuring blockchain parameters, such as block timeout, without compromising performance. DeLoRaN's scalability and efficient resource utilization, even in a fully connected mesh network, pave the way for further optimizations and highlight the importance of tailored blockchain configurations

to maximize efficiency in decentralized networks.

7.4 Comparison of DeLoRaN with HyperLoRa

In this section, we focus on comparing the performance of DeLoRaN with the study presented by Lu et al. [69] on a partially decentralized system, *HyperLoRa*, which integrates blockchain and edge computing in a LoRaWAN network. While HyperLoRa maintains a centralized NS for handling key operations like the deduplication of packets and forwarding payloads to Application Servers (ASs), it completely decentralize other operations like the join procedure. It is crucial that the join procedure in HyperLoRa is fully decentralized, as this allows us to finally compare DeLoRaN with another system. Most approaches analyzed in the state of the art have always retained some centralized component in key procedures, making a direct comparison with DeLoRaN difficult. HyperLoRa, however, presents a fully decentralized join procedure, providing us with a meaningful point of comparison for DeLoRaN's decentralized architecture. Additionally, both solutions use Hyperledger Fabric for blockchain, making it possible to compare their performance under similar conditions. Given that both systems address similar challenges, HyperLoRa provides an ideal benchmark to evaluate the scalability and efficiency of our fully decentralized solution.

7.4.1 Description of the experiments

The experiment focuses on testing the scalability and performance of DeLoRaN in comparison to HyperLoRa. Specifically, it evaluates how well both systems handle the join procedure under increasing network load, which is a critical operation in LoRaWAN systems. The test environment simulates a variable number of EDs, each attempting to join the network using a uniformly distributed wait time between join requests.

The experiment was designed to evaluate the join procedure scalability of DeLoRaN and compare it to HyperLoRa, focusing on response times under increasing load. However, due to limitations in the available data, CPU and RAM usage for HyperLoRa could not be evaluated, leaving us with a direct comparison based only on join procedure performance. Despite Lu et al. [69] provided details on such metrics, they were collected during other experiments involving the centralized NS, which makes impossible a fair comparison between the experiments.

The simulation setup included four HyperLoRa GWs (NCs in DeLoRaN) and a variable number of EDs, each generating join requests at randomized intervals. The traffic model utilized for both setups involved uniformly distributed wait times between 10 and 120 minutes for each join request.

The experiment was conducted using LXD containers for all NCs and Orderers, with each NC being hosted in its own container to create a realistic and manageable simulation environment. These containers mimicked a Smart6818 board, with its Samsung S5P6818 octa-core Cortex-A53 CPU, by heavily limiting the amount of CPU time used by the containers, and allowing each one to use just 1GB of RAM, which is roughly comparable to a Raspberry Pi 4's performance. The connection between containers was capped at 100MB, providing a constrained but realistic environment for simulating the limitations of low-end devices in the real world.

By using this setup, we aimed to test the scalability of both systems, focusing on the efficiency of the join procedure under increasingly heavy network loads while ensuring fair conditions for both DeLoRaN and HyperLoRa.

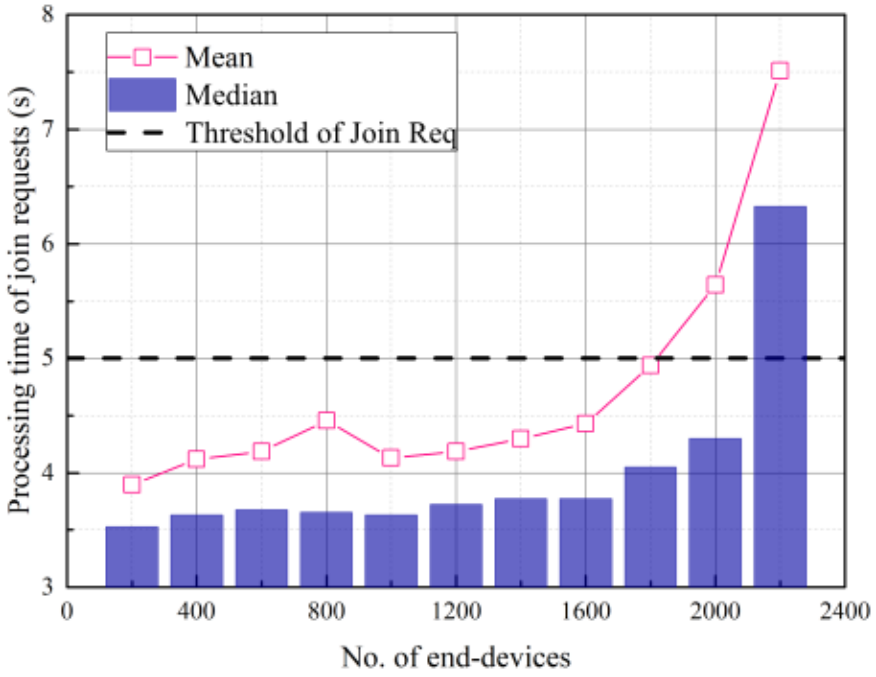


Figure 7.7: HyperLoRa delays with increasing devices number. It shows the response times of the join procedure as the number of devices increases exponentially. The response times increase sharply after 2000 devices, indicating that the system struggles to handle large numbers of join requests efficiently. Extracted from Lu et al. [69].

7.4.2 Results

HyperLoRa Presented Data – The results from the study conducted by Lu et al. [69] highlight significant scalability limitations in their partially decentralized HyperLoRa system. As illustrated in Figure 7.7, extracted from their paper, the join procedure begins to degrade after the number of end devices exceeds approximately 1800. At this point, around 25% of the end devices are unable to join the network due to delayed responses. Beyond the 2000-device threshold, the majority of devices fail to complete the join process successfully, due to HyperLoRa GWs inability to answer during the correct downlink window, as a result of the overloaded network.

This finding emphasizes the performance bottleneck encountered in HyperLoRa's design, where the system struggles to handle large numbers of join requests efficiently. The processing time for these requests, as shown in the Fig.7.7, increases sharply after the number of devices exceeds 2000, suggesting that the network infrastructure, even with partial decentralization, becomes overwhelmed. This serves as a valuable reference for assessing the scalability of DeLoRaN and understanding the critical importance of well coordinated decentralized architectures for managing large-scale IoT networks.

DeLoRaN Results – Our experimental results, presented in Fig. 7.8 showcase DeLoRaN's ability to manage join requests under heavy load, significantly outperforming HyperLoRa. Using the same traffic model, our system successfully handled 17.5 times more join requests than HyperLoRa, while keeping the response times well within acceptable limits. This marked improvement is particularly notable given the scalability challenges typically encountered in LoRaWAN networks.

Even with a vastly increased number of devices, DeLoRaN maintained an average response time of less than 20% of what was reported in HyperLoRa's experiment. This dramatic reduction in

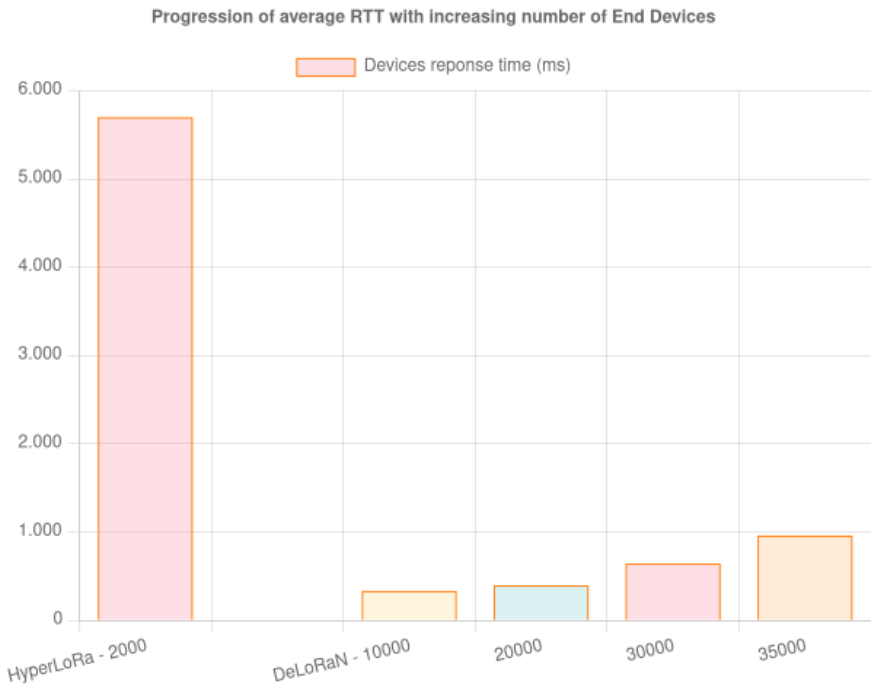


Figure 7.8: DeLoRaN delays with increasing devices number. It shows the response times of the join procedure as the number of devices increases exponentially. The response times increase slightly with the increasing number of devices, showing DeLoRaN's ability to handle tents of thousands of devices with a realistic traffic model.

response times, despite the higher load, underscores the superior efficiency and scalability of our fully decentralized architecture. It highlights DeLoRaN's ability to manage large-scale LoRaWAN networks with increased reliability, even when deployed on resource-constrained devices.

Moreover, the analysis of CPU and RAM usage from our experiments with DeLoRaN shows consistent and manageable resource consumption, even as the number of devices increases. As depicted in the plots, CPU usage increases steadily with the rise in the number of devices, reaching approximately 22-23% when scaled up to 35,000 devices. This gradual increase highlights that even at larger scales, DeLoRaN manages to distribute computational tasks efficiently across its decentralized architecture, preventing any significant spikes in CPU demand.

Similarly, RAM usage too follows a slight proportional increase, with consumption rising from around 300MB with 10,000 devices to about 450MB at 35,000 devices. These values demonstrate that DeLoRaN is able to handle a substantial number of connections while maintaining efficient memory management. This is particularly notable given the constraints of the simulated devices, further emphasizing the efficiency of our decentralized solution in low-resource environments.

Overall, DeLoRaN scales efficiently in terms of resource utilization, making it well-suited for large-scale IoT deployments, particularly where CPU and memory constraints are present, such as in edge computing scenarios. The slight linear increase in resource usage is a promising indication of the system's capacity to handle further scaling without overwhelming the network infrastructure.

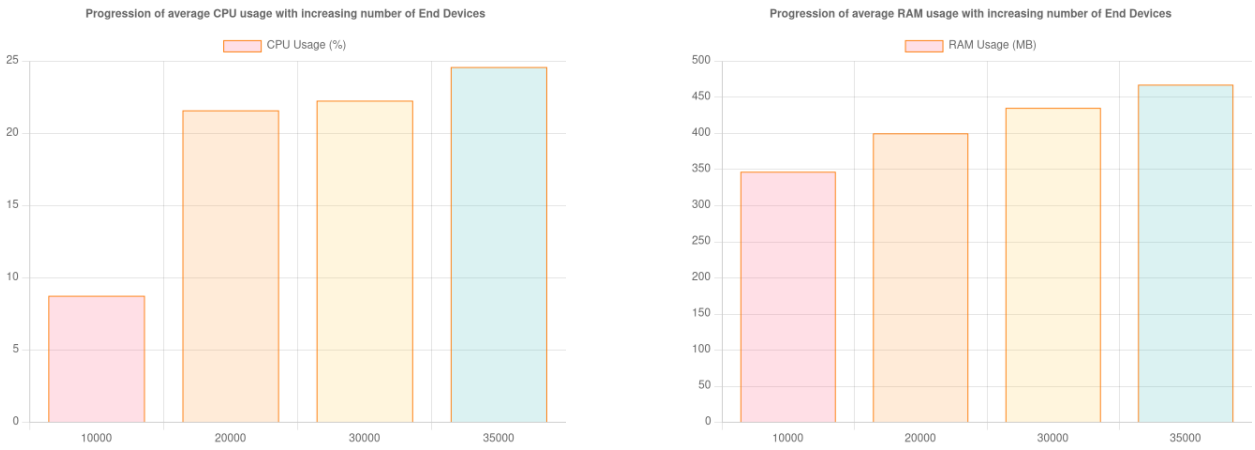


Figure 7.9: DeLoRaN RAM utilization with HyperLoRa traffic model. It shows the CPU and RAM usage of the LXD container when running the DeLoRaN setup with the HyperLoRa traffic model.

7.5 Real-world Data Set - LoED

The LoED dataset, presented in the work by Bhatia et al. [70], is a comprehensive dataset designed to capture real-world traffic patterns of LoRaWAN devices. It focuses on providing an accurate representation of data transmission behaviors at the edge, particularly useful for studying LoRaWAN in the context of large-scale deployments. The dataset was collected by observing the behavior of multiple devices over extended periods, focusing on the interaction between end devices and gateways in a realistic network environment. LoED contains data related to join requests, uplink, and downlink transmissions, with precise timestamps and metadata on packet reception, signal quality, and other relevant communication metrics.

In our simulations, presented in [71], the LoED dataset served as the foundation for modeling LoRaWAN traffic. The dataset's recorded traffic patterns allowed us to generate realistic device behavior, simulating thousands of end devices transmitting and receiving data in a distributed LoRaWAN architecture like DeLoRaN. To ensure accuracy, we used the dataset to model key characteristics, such as packet inter-arrival times and distribution of uplink and downlink messages.

Our traffic modeling approach aimed to reflect the complexity of real-world conditions. For instance, we incorporated variations in packet transmission rates and account for network congestion. Devices were modeled based on observations of their sending patterns and how they manage multiple packets during their active periods. We differentiated between "regular" and "non-regular" devices, allowing us to generate traffic with variable inter-arrival times, ensuring the simulation captured both periodic and aperiodic communication behaviors.

7.5.1 Traffic Modelling

We started our analysis on the previously introduced LoED dataset. For our purposes, we considered devices which have sent more than one packet. Let $\mathbb{D}$ be the set of all EDs and let D be its cardinality. Computing D, despite being a seemingly straightforward operation, presents challenges. Due to the structure of LoRaWAN packets, we only have DevAddr identifiers that may change over time. Therefore, we can only observe and compute the number of devices active simultaneously.

For each device $d \in \mathbb{D}$, we define

$$\mathbb{F}_d \;=\; \big\{\, i \in \mathbb{N} \;\big|\; \text{a packet } p_i^d \text{ with FCnt } = i \text{ was observed} \big\}.$$

Hence, $\mathbb{F}_d$ is the set of all observed frame-counter indices for device d. We then define the global set of all frame-counter indices as

$$\mathbb{F} \;=\; \bigcup_{d \in \mathbb{D}} \mathbb{F}_d.$$

Next, we let t_i^d be the timestamp of reception of packet p_i^d. We can thus define the collection of all timestamps as:

$$\left\{ t_i^d \right\}_{d \in \mathbb{D}, i \in \mathbb{F}} \tag{7.1}$$

Let $n(t)$ be the number of devices which are active at time t. A device is active at time t if its first packet is observed before time t and its last packet is observed after time t. We define N as the number of the maximum number of devices which are active at the same time. On our data we find $N \approx D = \max_t n(t) = 1521$.

During evaluation runs, we will use this value as the number of devices to simulate.

We start modelling the traffic by analyzing the inter-arrival times of packets. The inter-arrival time $\bar{t}_i^d$ is the time elapsed between two consecutive packets sent by the same device d. Let us denote a LoRaWAN packet as p and the time that a packet is sent as $t(p)$. Moreover, we indicate with p_i^d the packet of device $d \in \mathbb{D}$ having LoRaWAN Frame counter, FCnt $= i$. The FCnt is a value sent as clear text in the LoRaWAN header increased by 1 for each transmitted packet. To investigate the inter-arrival times of device d, we isolate the packets transmitted by the same device and compute the time elapsed by two consecutive packets:

$$t_i^d = t\left(p_{i+1}^d\right) - t\left(p_i^d\right) \tag{7.2}$$

It is important to note that the inter-arrival should be computed only on packets having consecutive values of FCnt. Indeed, in the measurements relevant to the dataset, several packets are lost on the air, causing missing observed FCnt values. In other words, it is impossible to compute t_i^d for some values of i since p_{i+1}^d may be lost and not observed.

For each device d, we computed its average inter-arrival time $\bar{t}^d = \text{avg}\left(\left\{t_i^d, \forall i\right\}\right)$. The distribution of $\bar{t}^d$ values is reported in the top plot of Fig. 7.10. We notice that this distribution significantly differs from Poisson or Pareto processes, which are commonly used in literature to describe LoRaWAN packet arrival rates. Indeed, such distribution presents some clear peaks, with two very high ones at 5 and 10 minutes. Other smaller peaks are observable, in particular the rightmost one located at 3600 seconds, i.e. one hour.

We started to investigate *regular* devices. A device is named *regular* if its inter-arrival times are constant throughout its lifespan, e.g., a device sending a packet continuously every 10 minutes. We calculated the average jitter as the variation of the inter-arrival times of the same device:

$$\bar{e}^d = \text{avg}\left\{ \left| t_i^d - t_j^d \right| \right\}_{d \in \mathbb{D}, i, j \in \mathbb{F}} \tag{7.3}$$

The distribution of the $\bar{e}^d$ values, for each device d, is reported in the bottom plot of Fig. 7.10. From

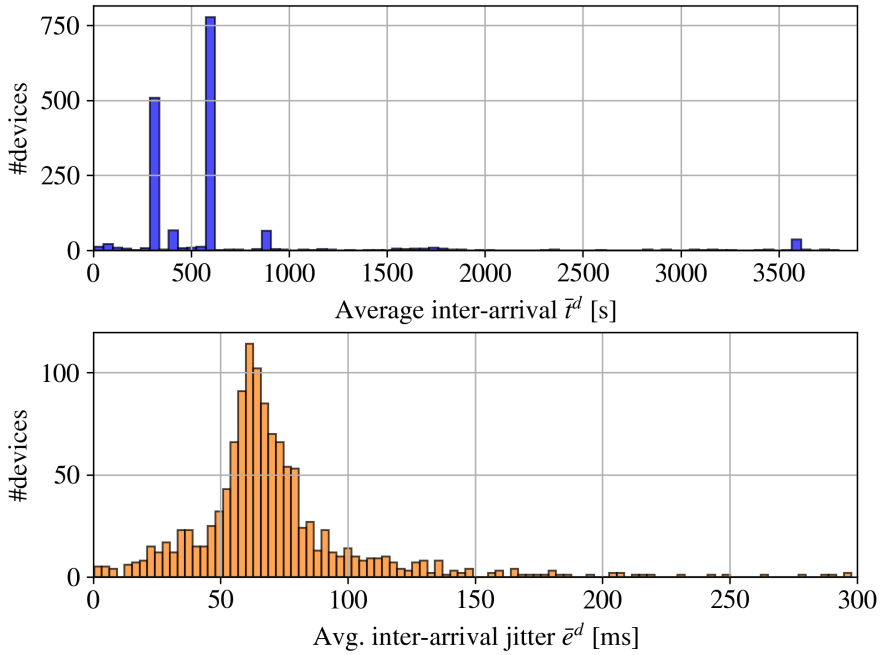


Figure 7.10: On the top plot, distribution of the average inter-arrival time for all EDs. The two highest peaks are located at 5 and 10 minutes respectively. on the bottom plot, distribution of the average inter-arrival jitter for all devices. The distribution presents a heavy tail not visible in the graph. The fraction of devices having $\bar{e}^d > 1$s is approximately 13% of the total.

the plot, we can see that the majority of devices are indeed regular, exhibiting a quasi-constant value of inter-arrival times and therefore a low value of $\bar{e}^d$. For our purposes, we consider a device d as regular if $\bar{e}^d < 1$s. We note that the value of 1s is much lower that the average inter-arrival time, as shown in the distribution in Fig. 7.10. Using a threshold of 1 second, we define the number of regular devices as $R = 1418$. Consequently, we define $\rho = 0.866$ as the fraction of regular devices relative to the total number of devices.

While the value of ρ is quite high, it is lower than one may expect by looking at Fig. 7.10. This is because the distribution presents a heavy tail, which is not visible in the figure. By separating regular and non-regular devices, we empirically model the probability distributions $P_r(t)$ and $P_{nr}(t)$, which represent the inter-arrival times of regular and non-regular devices, respectively.

Device simulation policy – We execute the following procedure to model the behaviour of a simulated LoRaWAN device d. First, we extract a random number $a \in [0, 1]$. If $a \leq \rho$ then d is regular. In this case, a single inter-arrival time t^d is drawn from the probability distribution $P_r(t)$, which will be the constant inter-arrival time of the device for its entire lifespan. If otherwise $a > \rho$ the device is not regular. In this case, whenever the device sends packet i, we draw time t_i^d from the probability distribution $P_{nr}(t)$ and this will be the time that it has to wait to send packet $i+1$.

7.5.2 Simulation results

The setup of the experiment is the same used in Sec. 7.4. We performed the simulations using the traffic model proposed in the previous section. In the simulations, the devices follow the following process: they first power on and complete their initial join procedure with the network. After successfully joining, the devices proceed to send uplinks according to the timings defined in the traffic model, reflecting realistic network behaviour. Each device emulates a LoRaWAN ED, generating

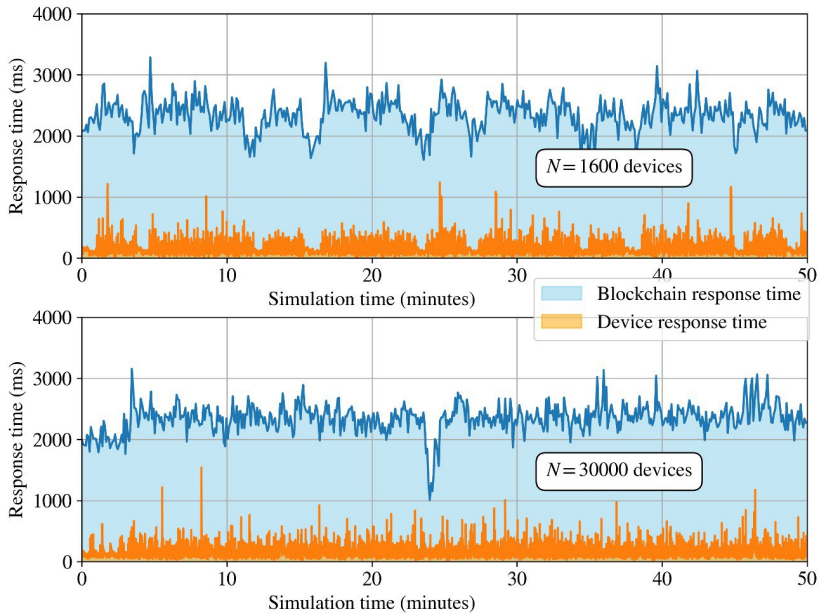


Figure 7.11: Blockchain Response time and ED Response time under LoED traffic model with 1600 EDs on the top and 30000 EDs on the bottom.

the exact same bytes as in the traffic model. During the experiments, the system ran overnight with the devices generating packets in real-time. The results presented here are based on 50-minute samples taken from several hours of simulation. We conducted the simulations with two different numbers of devices. First, we used 1600 devices, which is the number of devices we derived from LoED dataset. The second number is much higher, around 30000 devices. This was chosen to perform a stress test on DeLoRaN and observe how the system behaves under a heavy load.

System Response Time – We started by measuring the system's response time. For our testing, we considered two types of response times: the blockchain response time and the device response time:

- **Blockchain Response** Time is the time between a transaction being sent and a confirmation from the blockchain network being received. It is measured from the NC by storing the timestamps between and after the transaction proposal process.
- **ED Response Time** is the time spent waiting between a confirmed uplink being sent and the respective ACK downlink being received. It is measured from the ED by storing timestamps when an uplink is sent and a downlink is received.

Fig. 7.11 shows the response time of DeLoRaN under two different load conditions: 1600 devices (top chart) and 30000 devices (bottom chart). The blue line represents the blockchain response time, while the orange line shows the device response time. We have then simulated 30000 devices with a cumulative packet rate of 1.5×10^5 packets/h. With the higher number of simulated devices, the average response times are slightly higher. However, we observed latency spikes that were not present with the lower number of devices. Despite this, these spikes always remain below the critical threshold of 5 seconds, which is the time defined by the LoRaWAN standard after which the device waits for a downlink. This indicates that the system is able to handle the increased load without significant performance degradation.

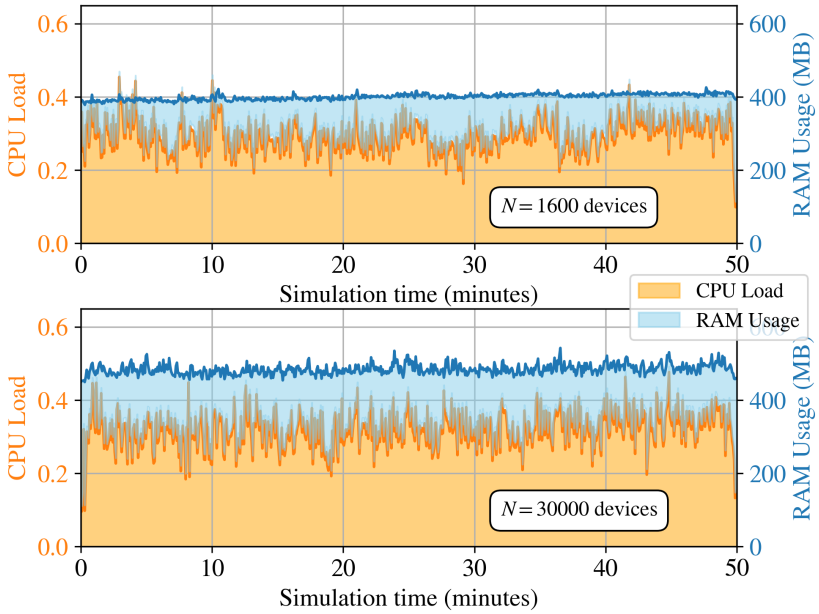


Figure 7.12: CPU and RAM utilization in a Network Controller under LoED traffic model with 1600 EDs on the top and 30000 EDs on the bottom. The results are averaged among all the Raspberry Pi nodes.

Resource utilization – In this section, we investigate the usage of computational resources by DeLoRaN. Fig. 7.12 displays the CPU load (in orange) and RAM usage (in blue) during simulations with 1600 devices (top) and 30000 devices (bottom) over 50 minutes. In both scenarios, the CPU load remains relatively stable throughout the simulation. With 1600 devices, the average CPU load hovers around 30%, with occasional spikes that remain well below 50%. In the 30000-device simulation, the CPU load increases slightly, averaging closer to 40%, with similar fluctuations. However, even under this significantly heavier load, the system stays well below saturation. Furthermore, it is important to highlight that these tests were conducted on a Raspberry Pi CPU, which is comparable to the CPUs used in many LoRaWAN GWs currently deployed worldwide. Therefore, the similarity in hardware adds validity to our tests, as it demonstrates that our system can perform efficiently under real-world conditions with similar computational resources. In terms of memory usage, the RAM consumption is also stable throughout the simulations. For both 1600 and 30000 devices, the RAM usage remains consistent, around 400/500 MB.

7.5.3 Sapienza Dataset

In addition to using the LoED dataset, which provides a comprehensive collection of LoRaWAN packet data, we are actively working on building a new dataset, shown in Fig. 7.13, to contribute to the relatively sparse set of publicly available LoRaWAN datasets. The scarcity of such datasets in the field of LoRaWAN research is a significant barrier, as most existing studies rely on limited or artificially generated data. Our project, which has already captured over 1.5 million packets from two LoRa GWs installed atop the DIET and DIAG buildings at Sapienza University, aims to address this gap. The primary goal is to create a valuable dataset for future researchers and to develop advanced noise and traffic models for LoRaWAN, specifically tailored to simulate urban environments more accurately. These models and datasets will be instrumental in enhancing the realism and accuracy of simulations in LoRaWAN research, ultimately advancing the state of the

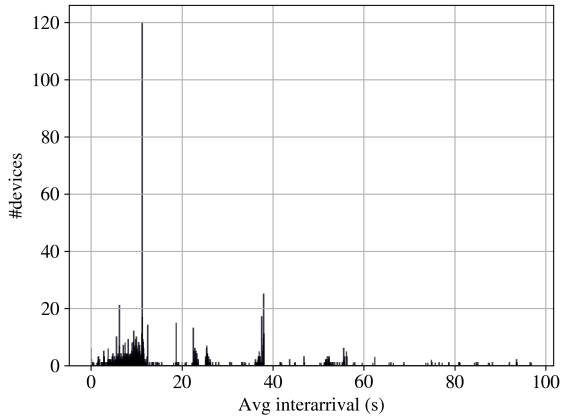


Figure 7.13: Sapienza dataset shows more frequent, shorter interarrival times when compared to LoED dataset, reflecting the dense urban environment in which it was collected.

art in urban IoT networks.

7.6 Conclusions

In this chapter, we have presented a comprehensive experimental evaluation of DeLoRaN, focusing on its performance, scalability, and resource efficiency when applied to decentralized LoRaWAN networks. Through a series of simulations, we have demonstrated that DeLoRaN significantly improves the handling of LoRaWAN traffic by distributing control across multiple NCs and utilizing blockchain to enhance security and reliability.

One of the key advantages of DeLoRaN, as demonstrated by our results, is its decentralized architecture. Traditional centralized systems, such as Chirpstack, rely on a single point of control, which can lead to bottlenecks, reduced scalability, and vulnerability to failures or attacks. In contrast, DeLoRaN eliminates these limitations by distributing the control logic across multiple NCs, reducing the risks associated with Single Point of Failures (SPoFs).

In contrast to centralized systems that often struggle with high device counts and traffic loads, DeLoRaN's decentralized structure demonstrated superior performance under heavy load, with the system efficiently distributing tasks among multiple NCs. This decentralized control not only reduces the computational and networking load on any single node but also improves fault tolerance and the overall reliability of the network. The system was shown to scale linearly with the number of devices, maintaining stable performance metrics, such as CPU usage and RAM consumption, throughout the experiments.

Our experiments confirmed that DeLoRaN is capable of scaling efficiently under varying conditions, maintaining low response times even as the number of connected devices increases exponentially. The system was able to handle up to 30,000 devices without suffering from performance degradation, proving that DeLoRaN is a robust solution for large-scale IoT deployments. Additionally, the ability to decouple blockchain processes from device communication allowed DeLoRaN to manage high loads without overwhelming network resources, leading to improved overall system performance.

The use of real-world traffic data from the LoED dataset provided an accurate basis for our

simulations, ensuring that our results reflect realistic operational conditions. DeLoRaN's traffic modeling, based on LoED data, allowed us to accurately replicate the behavior of LoRaWAN devices in various environments, including the challenges posed by urban settings. By leveraging this dataset, we were able to identify key patterns in packet transmission, inter-arrival times, and response times, all of which informed the refinement of DeLoRaN's architecture.

Furthermore, our resource utilization analysis showed that DeLoRaN's decentralized architecture is well-optimized, with CPU usage remaining stable and RAM consumption within acceptable limits, even under stress tests involving 30,000 devices. This efficiency makes DeLoRaN suitable for deployment on resource-constrained hardware such as Raspberry Pi-based gateways, ideal for real-world IoT implementations.

In conclusion, DeLoRaN represents a major advancement in the management of decentralized LoRaWAN networks, offering a scalable, resilient, and efficient solution for modern IoT applications. Its architecture not only addresses the limitations of centralized systems but also paves the way for further enhancements, particularly in optimizing blockchain parameters to fully exploit it. Future work will focus on refining the system to accommodate the growing demands of IoT ecosystems and ensuring continued advancements in the field of decentralized network management.

Chapter 8

Smart Jamming/Interference Detector for LoRaWAN

8.1 Introduction

The Adaptive Data Rate (ADR) mechanism in LoRaWAN is designed to optimize network efficiency by dynamically adjusting key transmission parameters, such as the Spreading Factor (SF), transmission power, and data rate, and is defined in the LoRaWAN standard [21]. By leveraging ADR, LoRaWAN networks aim to strike an optimal balance between communication reliability, energy efficiency, and spectral efficiency, enabling end devices to maintain extended battery life while ensuring effective communication over large areas. This mechanism is particularly important for Low-Power Wide-Area Network (LPWAN) applications that require long battery life, as in smart cities, agriculture, and logistics. However, despite its advantages, the ADR algorithm has several inherent limitations, especially in dynamic environments with sudden changes in radio conditions, such as those caused by interference and jamming attacks.

Algorithm 6 ADR Algorithm

```
1: ADR_ACK_CNT = 0
2: ADR_ACK_DELAY = 32
3: ADR_ACK_LIMIT = 64
4: if uplink transmission
5:    ADR_ACK_CNT = ADR_ACK_CNT + 1
6:    if ADR_ACK_CNT == ADR_ACK_LIMIT
7:       Send a confirmed uplink message
8:    end if
9: end if
10: if ADR_ACK_CNT ≥ ADR_ACK_LIMIT + ADR_ACK_DELAY
11:    increase SF and Tx Power
12: end if
13: if received downlink transmission
14:    ADR_ACK_CNT = 0
15: end if
```

The current implementation of the ADR algorithm, described in Alg. 6, relies heavily on historical link quality indicators, such as the Signal to Noise Ratio (SNR) and Received Signal Strenght

Indicator (RSSI). These indicators are used to adjust the network parameters gradually, which means that ADR operates in a reactive manner. It attempts to optimize device settings based on average link quality over relatively long periods. While this design approach is well-suited for relatively stable environments, it becomes a significant drawback when link conditions change abruptly [72], as in the case of interference or deliberate jamming. When faced with sudden degradation in link quality, ADR's response is inherently delayed, as it relies on accumulating sufficient link quality information over multiple transmissions before making any adjustments. This delayed response means that devices may continue using non-optimal settings for an extended period, which leads to an increased likelihood of packet loss, wasted energy, and ultimately poor network performance.

Interference from other radio signals and unintentional noise are common occurrences in shared spectrum environments like those used by LoRaWAN [73]. In such situations, the reactive nature of ADR results in inefficiencies. Since ADR takes time to detect and adjust to changing conditions, the end devices may transmit with an SF or power level that is no longer ideal, causing an increase in retransmissions and energy consumption. In dense deployments, where multiple devices operate in close proximity and compete for limited spectrum resources, these inefficiencies are further amplified, leading to a decrease in overall network throughput and reduced Quality of Service (QoS).

A particularly challenging scenario for ADR is when LoRaWAN networks face deliberate jamming attacks [74]. Jamming occurs when an attacker intentionally emits radio signals in the same frequency band used by the network, with the objective of creating interference that disrupts legitimate communications. Jamming can be carried out in a variety of ways, such as continuous jamming, where the attacker transmits constantly, or reactive jamming, where the attacker only transmits when detecting an active communication. In either case, the ADR algorithm in its current form struggles to adapt effectively. Because ADR relies on the average SNR over many transmissions, it fails to differentiate between transient jamming events and actual long-term link quality changes. As a result, ADR may either react too slowly or fail to react at all, leaving end devices unable to maintain stable communication.

Furthermore, the ADR algorithm's optimization strategy is focused on gradually reducing the SF and transmission power to minimize energy consumption when link quality is good. While this approach is beneficial under normal conditions, it becomes counterproductive during jamming attacks. When a jamming signal is present, the link quality degrades rapidly, but ADR may still try to reduce power or SF based on outdated link quality information, further exacerbating packet losses. The lack of real-time adaptability means that, in the presence of jamming, ADR not only fails to mitigate the attack but may also inadvertently contribute to increased channel congestion [75] by forcing retransmissions, thereby reducing network efficiency and reliability.

The problem is further complicated by the distributed nature of LoRaWAN networks. Unlike traditional centralized cellular networks, where a base station has complete information about all connected devices, LoRaWAN relies on distributed Gateways (GWs), each of which has limited visibility of network conditions due to its simplicity. This distributed architecture means that individual GWs just do not have sufficient information to detect jamming or other forms of interference quickly. The ADR mechanism, which relies on data from both the end devices and GWs, is thus limited in its ability to make informed decisions in real-time. Consequently, the lack of coordination and real-time awareness across GWs results in inconsistent responses to network impairments, further reducing the effectiveness of ADR in mitigating interference and jamming.

In this context, it is evident that the existing ADR mechanism is ill-equipped to handle the challenges posed by sudden interference or malicious jamming attacks. The need for a more resilient approach is critical to ensure reliable communication in dynamic and potentially hostile environments. A decentralized and more adaptive ADR strategy, capable of responding rapidly to changes in link conditions, is necessary to address these shortcomings. Such an approach would ideally incorporate real-time feedback mechanisms, allowing for quicker adjustments in transmission parameters, thereby enhancing the robustness of LoRaWAN networks against both unintentional interference and deliberate jamming.

This chapter explores a jamming/interferences detection and mitigation technique in LoRaWAN networks, which is ideally intended to be part of a larger, decentralized smart ADR approach. The proposed solution, presented in [76], aims to effectively detect and mitigate jamming attacks or stable interferences, while also setting the foundation for an adaptive, decentralized mechanism that enhances network resilience, efficiency, and QoS in dynamic environments.

8.2 Background and State of the Art

The ADR mechanism in LoRaWAN aims to balance the communication efficiency and power consumption of end devices. By dynamically adjusting the SF, transmission power, and data rate, ADR ensures that end devices transmit at the lowest possible power while maintaining reliable communication with GWs. The ADR algorithm is particularly effective in stable environments where link quality changes slowly over time. It optimizes the use of network resources by allowing devices closer to gateways to transmit using lower SFs, which reduces the Time on Air (ToA) and thus increases network capacity.

However, ADR's reliance on historical link quality metrics makes it inherently reactive rather than proactive. When the network experiences rapid changes in link quality due to interference or jamming, ADR struggles to respond effectively. Research by Li et al. [77] demonstrated that the current ADR mechanism often exacerbates performance issues during periods of interference. They found that ADR's inability to adapt to rapid changes in link conditions results in increased retransmissions and energy consumption, especially in dense network deployments. This limitation highlights the need for an enhanced ADR mechanism capable of responding in real-time to changing link conditions, particularly under adversarial scenarios such as jamming. For example, when a jamming attack occurs, the SNR drops abruptly, but ADR continues to use outdated link quality data, leading to a delayed response. As a result, end devices may continue transmitting at suboptimal settings, resulting in poor communication reliability.

8.2.1 Related Works

Several studies have focused on enhancing the ADR mechanism to improve LoRaWAN's performance under dynamic conditions.

Xu et al. [78] discusses detecting jamming attacks in wireless sensor networks using statistical metrics such as signal strength, carrier sensing time, and packet delivery ratio (PDR). It proposes a multimodal consistency check, combining these metrics to improve detection accuracy. The basic metrics are easy to implement but can lead to false positives due to normal network issues, while the multimodal approach is more reliable but adds complexity and increases energy consumption.

Magrin et al. [79] discuss that interference in LoRaWAN, particularly due to overlapping transmissions, significantly impacts network performance, leading to increased packet loss and reduced reliability. The ADR mechanism is criticized for its inefficiency under dynamic conditions, where rapid changes in the environment make it slow to adapt. The paper discusses possible improvements to ADR, including more dynamic adjustments to spreading factor and power levels, aimed at enhancing scalability, fairness among nodes, and resilience to changing channel conditions

One of the major concerns is the inherent vulnerability of LoRaWAN's slow modulation scheme to reactive jamming attacks, which can be executed using low-cost hardware. Huang et al. [80] experimentally evaluates the vulnerability of LoRaWAN to jamming attacks, emphasizing the non-orthogonality of transmissions as a key weakness. It utilizes reactive jamming with Channel Active Detection (CAD) to effectively disrupt communication, showing that LoRaWAN's ADR mechanism cannot fully mitigate these vulnerabilities. The study highlights the need for enhanced countermeasures to address the susceptibility of LoRaWAN to interference and jamming.

Similarly, Martinez et al. [81] evaluates the impact of jamming on LoRaWAN, highlighting significant vulnerabilities to both channel-aware and channel-oblivious jammers. It presents a performance analysis of LoRaWAN under both channel-aware and channel-oblivious jamming conditions, showing a 56% reduction in network throughput. It demonstrates that ADR is ineffective against these types of jamming, leading to severe performance degradation. The study calls for enhanced countermeasures, focusing on the need to address these vulnerabilities to maintain reliable network performance under adversarial conditions.

Ruotsalainen et al. [82] reviews jamming attacks on LoRaWAN, emphasizing the susceptibility of the physical layer to different types of jamming, including reactive RF jamming and triggered jamming. These attacks can significantly degrade network performance by causing packet loss and increased energy consumption. The study also discusses wormhole attacks and Denial-of-Service (DoS) as threats that manipulate physical-layer vulnerabilities to disrupt communication.

For jammer detection, the paper outlines approaches using statistical methods (like Kullback-Leibler divergence) and Recurrent Neural Networks (RNNs) to identify anomalies based on network metrics, such as RSSI and packet inter-arrival times. These methods help in distinguishing jamming activities from normal network behavior.

The paper also critiques the ADR mechanism of LoRaWAN, noting that its effectiveness is limited under dynamic jamming conditions. Attacks can exploit ADR, forcing suboptimal settings, resulting in degraded communication reliability and increased energy usage. The authors highlight the need for enhanced ADR strategies and robust countermeasures to improve network resilience against these attacks.

Ingham et al. [83] present an in-depth analysis of predictive jamming in LoRaWAN, highlighting its effectiveness and proposing an Intrusion Detection System (IDS) based on predictive signal analysis to detect these sophisticated jamming attacks. The IDS targets the predictability of LoRaWAN, distinguishing between normal behavior and deliberate jamming. The paper also critiques ADR for its lack of adaptability in responding to dynamic interference.

Magrin et al. [84] focus on LoRaWAN's performance under interference in urban environments, evaluating the limitations of ADR when faced with high-density networks and overlapping signals. The analysis suggests that ADR is often ineffective in such scenarios, leading to performance degradation. No specific jamming detection mechanism is proposed, but the paper emphasizes the need

for more adaptive ADR methods to handle real-world conditions.

The existing body of research indicates a growing recognition of the need to enhance LoRaWAN's security posture through both technological and algorithmic advancements, especially in the face of low-cost, accessible jamming hardware and the increasing prevalence of network-based attacks. Building on existing literature, this thesis integrates machine learning techniques for real-time attack detection and proposes a novel approach for direct mitigation at the GW, thereby enabling decentralized management and improving the resilience of LoRaWAN networks against various jamming threats.

8.3 Scenario Model

In order to thoroughly evaluate the vulnerabilities and resilience of LoRaWAN under adversarial conditions, it is crucial to model both the network scenario and the attacker accurately within a simulation environment. By simulating the behavior of legitimate devices and potential attackers, we can assess the impact of jamming attacks and interference under controlled, repeatable conditions, providing valuable insights into system performance and mitigation strategies.

NS-3[1] is an open-source, discrete-event network simulator widely used for research in networking. It provides a flexible and customizable platform for simulating complex communication scenarios, allowing detailed modeling of protocols, interference, and attacker behaviors, making it ideal for assessing LoRaWAN performance under different attack models.

The study comprises three distinct simulation scenarios: normal traffic, channel-oblivious jamming, and channel-aware jamming, each designed to comprehensively assess the performance of LoRaWAN networks under varying conditions of interference. These scenarios help illustrate the system's capabilities and vulnerabilities when operating under both benign and hostile environments, which are crucial for understanding how real-world deployments can be affected by malicious actors.

Normal Traffic Scenario – This scenario represents the baseline operation of a LoRaWAN network under ideal conditions, without any external interference. The GW receives packets from End Devicess (EDs) that are evenly distributed across a 12 km by 12 km area, providing insights into the stability and performance of LoRaWAN communication without any disruptions. This setup allows us to measure the expected packet success rate, Inter-Arrival Time (IAT), Packet-Delivery Ratio (PDR) and the overall efficiency of the network when all components work as intended, serving as a reference point for evaluating the impact of external jamming attacks.

Channel-Oblivious Jamming Scenario – The channel-oblivious jamming scenario introduces a jammer that operates in randomly timed cycles, with both active and inactive periods having random durations. This type of jammer does not adapt to specific network conditions but instead transmits at arbitrary times, generating generalized, non-targeted interference. The result is a more dispersed, unpredictable disruption of communication that moderately impacts the network's performance. The channel-oblivious jammer provides insights into how LoRaWAN behaves under basic jamming conditions where interference is unsystematic but still capable of degrading network reliability. By analyzing this scenario, we can better understand the resilience of the network to non-coordinated jamming activities and evaluate how the overall PDR and IAT are affected.

[1] NS-3 - https://www.nsnam.org/

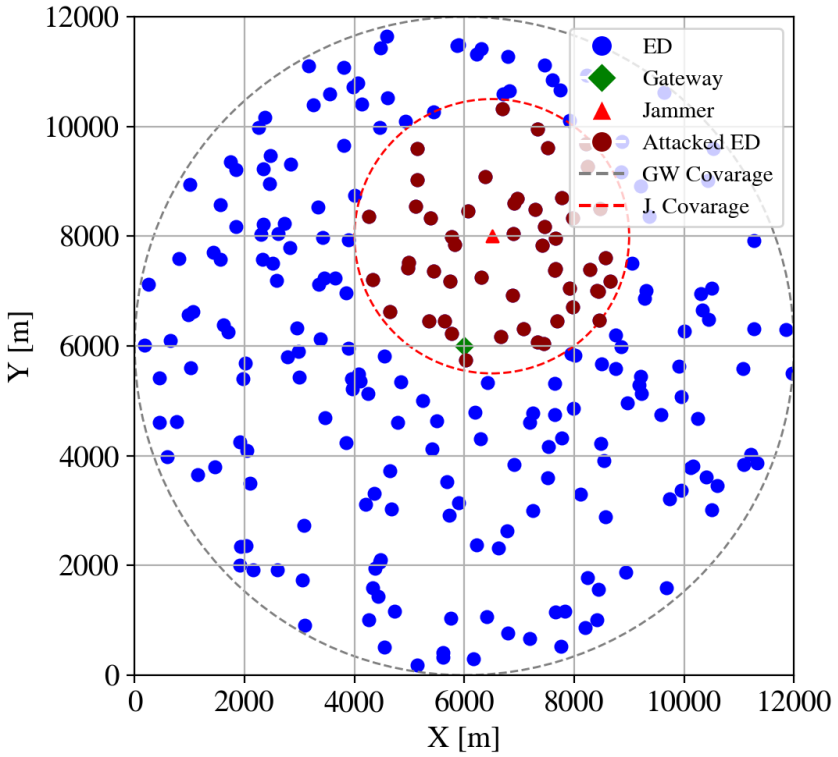


Figure 8.1: Network Topology. The simulation setup consists of 250 EDs distributed across a 12 km by 12 km area, a GW, and a jammer.

Despite the simplicity of this attack model, it provides valuable insights into the vulnerabilities of LoRaWAN under basic jamming conditions, highlighting how an attacker with basic capabilities can still disrupt the efficiency of our network.

Channel-Aware Jamming Scenario – The channel-aware jamming scenario presents a more sophisticated approach, employing a smart jammer capable of actively monitoring the network traffic to identify the channels and SFs in use by the EDs. By listening to network activity, the jammer can determine when and where legitimate transmissions are taking place, allowing it to send interfering packets on the same channel and SF as the targeted EDs. This type of attack is far more effective because it directly disrupts legitimate transmissions at precise moments, significantly reducing the number of packets successfully received by the GW. This scenario simulates an advanced attacker that uses intelligent interference to maximize disruption and provides critical insights into the vulnerabilities of LoRaWAN when an adversary can dynamically adjust their jamming strategy based on real-time network conditions. The performance under such attacks is measured by the number of successfully received packets and the resulting IAT, which reflects the impact on communication stability and timing, highlighting how sophisticated jamming can degrade network reliability.

8.3.1 Network Model

Fig. 8.1 illustrates the spatial layout of the simulation setup, featuring 250 EDs distributed across a 12 km by 12 km area, a centrally positioned GW, and a single jammer. The large-scale deployment aims to mirror realistic LoRaWAN scenarios, with devices spread evenly throughout the area to emulate the typical coverage conditions and transmission dynamics seen in actual Internet of Things (IoT) networks. This configuration allows for a comprehensive examination of network behavior

under both normal and adversarial conditions, ensuring that the results are representative of the performance and vulnerabilities of LoRaWAN in large, geographically dispersed deployments.

8.3.2 Threat Model

The threat model focuses on demonstrating the vulnerability of LoRaWAN networks to jamming attacks even when conducted by an unsophisticated adversary with minimal resources. The attacker is assumed to have very basic hardware, such as an off-the-shelf Long Range (LoRa) radio module, and does not employ advanced jamming techniques, making the threat realistic for low-effort, low-cost attackers. This scenario aims to illustrate that even a "lazy" or low-skilled attacker can effectively disrupt communication within a LoRaWAN deployment, requiring very little technical expertise or investment.

The attacker uses an ordinary LoRa radio that is widely available and inexpensive, highlighting that access to specialized or sophisticated jamming tools is not required. This basic jammer operates by transmitting packets on the same frequencies and using similar SFs as the legitimate EDs, thereby interfering with normal communications. The threat model encompasses two types of jamming:

- **Channel-Oblivious Jamming**: The attacker transmits jamming signals randomly, without analyzing network activity. This non-targeted interference still disrupts LoRaWAN communication, causing packet loss and performance degradation.
- **Channel-Aware Jamming**: The attacker listens for preambles to identify active channels and SFs, then transmits at the right moments to interfere. Even with basic hardware, this targeted approach effectively prevents packet reception at the GW.

The main aspect of this threat model is the low barrier to entry for launching a successful jamming attack. By using a readily available LoRa radio, the attacker can cause considerable degradation to network performance with minimal effort. This underscores the vulnerability of LoRaWAN networks to interference, where even basic tools can be used to carry out highly effective attacks. The simplicity and affordability of the attack setup emphasize the importance of strengthening LoRaWAN's resilience to jamming, as the potential threat does not come solely from sophisticated adversaries but also from anyone with access to basic LoRa equipment.

The threat model reveals that even an unskilled attacker, equipped with nothing more than a low-cost LoRa transceiver, can cause significant disruption by either transmitting continuously or selectively after listening to active communication. This highlights the need for enhanced security measures and robust jamming detection and mitigation strategies to ensure that LoRaWAN networks can remain operational and reliable, even in the presence of such low-effort adversaries.

8.3.3 Defensive Model

We need ways to mitigate the impact of jamming on LoRaWAN networks, which are usually classified into three categories: Detection Techniques, Proactive Countermeasures, and Reactive Countermeasures, similar to the approach taken in [85] for Wireless Sensor Networks (WSNs).

Detection Techniques – Detection techniques focus on identifying the presence of a jammer in the network through various detection strategies. Detection can be achieved through statistical methods or machine learning models, which analyze network metrics to identify anomalies indicative

of jamming. These techniques play a crucial role in initiating countermeasures to maintain network performance and security.

Proactive Countermeasures – Proactive countermeasures aim to prevent jamming attacks before they occur by enhancing the network's resilience. The key mechanism employed here is Frequency Hopping. To enhance network robustness against interference and jamming, LoRaWAN supports this proactive countermeasure, where devices switch frequencies during transmissions. This technique makes it difficult for a jammer to consistently block communication since the devices are dynamically changing channels. By distributing transmissions across a wide range of frequencies, frequency hopping ensures that a single frequency jam does not entirely disrupt communication.

Reactive Countermeasures – Reactive countermeasures are deployed after jamming activity has been detected. These strategies focus on minimizing the impact of ongoing attacks by adapting network parameters and rerouting traffic. Examples include:

- **Jammer-Aware ADR**: This technique adjusts the data rate and transmission power of devices based on detected jamming conditions. By dynamically adapting the network parameters, devices can optimize communication to minimize the impact of the jammer.
- **Mapping Jamming Areas**: This involves identifying and mapping the areas affected by jamming attacks. By detecting the jammer's range and coverage, the network can reconfigure itself to avoid the impacted regions, redirecting traffic through unaffected areas.
- **External Sensing Nodes**: Deploying additional external nodes dedicated to sensing jamming signals can help monitor the network for interference. These nodes act as early warning systems, detecting jamming activity and alerting the network to take appropriate countermeasures.

Detection Techniques – The primary goal of detection techniques is to quickly identify jamming activity, making their implementation crucial at the lower levels of the network stack. When combined with other countermeasures, these methods significantly enhance the network's resilience. Useful detection techniques analyzed are:

- **Statistical Approaches**:
  - Z-Score;
  - Interquartile Range (IQR);
  - Mahalanobis Distance;
- **Machine Learning Approaches**:
  - Classification algorithms: decision tree, k-nearest neighbors (KNN), random forest;
  - Reinforcement learning techniques: Q-learning, deep Q-networks (DQN);

These detection algorithms rely on data collected from network components such as EDs, GWs, or Network Servers (NSs), or from external nodes specifically deployed for monitoring. We analyzed different metrics like:

1. **Average IAT**: Time interval between two consecutively received frames, recorded at the GW.

2. **RSSI**: Signal strength monitored at the GW.
3. **SNR**: Ratio between signal strength and background noise.
4. **Packet Success Rate (PSR)**: Number of successfully received packets, indicating network reliability.
5. **Packet Loss (PL)**: Difference between packets sent and received, indicating communication issues.

Studies such as Lagat et al. [86] and Richarson et al. [87] have demonstrated the effectiveness of machine learning models in detecting jamming and collision attacks with high accuracy by analyzing key network metrics like RSSI, SNR, and packet loss rate. These models significantly improve the network's resilience by enabling proactive detection and immediate mitigation, enhancing overall security under adversarial conditions.

8.4 Jammer Impact

8.4.1 Evaluation Metrics

The performance evaluation of the LoRaWAN network under jamming scenarios is based on several key metrics that provide insights into the network's efficiency, reliability, and resilience. These metrics help quantify the impact of jamming on network performance and assess the effectiveness of countermeasures in mitigating the effects of interference. The primary evaluation metrics include: PSR, IAT, Throughput, and Jitter.

Packet Success Rate – PSR represents the proportion of packets that are successfully transmitted without loss, making it a key metric for assessing network reliability, especially in jamming scenarios. PSR is defined as:

$$\text{PSR} = \frac{P_{\text{successful}}}{P_{\text{total}}} \tag{8.1}$$

where $P_{\text{successful}}$ is the number of successfully received packets, and P_{total} is the total number of transmitted packets.

Throughput – Throughput measures the rate of successfully received data over time, typically expressed in bytes per second. It indicates the network's efficiency in data delivery, particularly when jamming degrades performance. Throughput is calculated as:

$$\text{Throughput} = \frac{B_{\text{received}}}{T} \tag{8.2}$$

where B_{received} is the total number of bytes successfully received, and T is the duration of the observation period.

Inter-Arrival Time – IAT is the time interval between the reception of consecutive packets. It provides insight into the regularity of packet arrivals and is useful for analyzing the effects of jamming on packet timing. IAT is given by:

$$\text{IAT} = t_n - t_{n-1} \tag{8.3}$$

where t_n is the arrival time of the current packet, and t_{n-1} is the arrival time of the previous packet.

Jitter – Jitter quantifies the variability in the IAT of packets. High jitter indicates significant fluctuations in packet delivery times, which can be a sign of network instability, possibly worsened by jamming activity. Jitter is calculated as the standard deviation of IAT values:

$$\text{Jitter} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(IAT_i - \overline{IAT})^2} \tag{8.4}$$

where IAT_i is the IAT of the i-th packet, $\overline{IAT}$ is the mean IAT, and N is the total number of packets.

These metrics together provide a comprehensive assessment of network performance under both normal and jamming conditions.

8.4.2 Normal Scenario Configuration

Before introducing the jammer, we model the network under normal traffic conditions to establish a performance baseline. This scenario represents a typical LoRaWAN deployment where EDs communicate with a single GW over predefined uplink frequencies.

- **Network Configuration**: The network consists of one GW and multiple EDs randomly distributed over a predefined area, operating on standard uplink frequencies (868.1, 868.3, 868.5 MHz).
- **End Device Operation**: Each ED transmits 23-byte packets at regular intervals, using the most suitable SF (SF7-SF12) based on its distance from the GW. All devices operate under Class A mode.
- **No External Interference**: There is no jamming or other interference affecting the network in this initial phase.

This setup serves as a reference point to evaluate the effects of the jammer in subsequent scenarios.

8.4.3 Jammer Configuration

The jammer operates on the same uplink frequencies as the EDs (868.1, 868.3, 868.5 MHz) with realistic transmission power but without duty cycle limitations, resulting in continuous jamming.

- **Random Activation and Deactivation**: The jammer sends a packet every 45 seconds and enters random idle intervals between 4 and 12 hours to simulate real-world unpredictability.
- **Slot-Based Activation**: During active periods, the jammer alternates between transmission and idle slots to conserve energy and avoid detection.
- **Dynamic Frequencies and Spreading Factors**: The jammer dynamically changes frequencies and SFs during each activation to increase interference complexity.

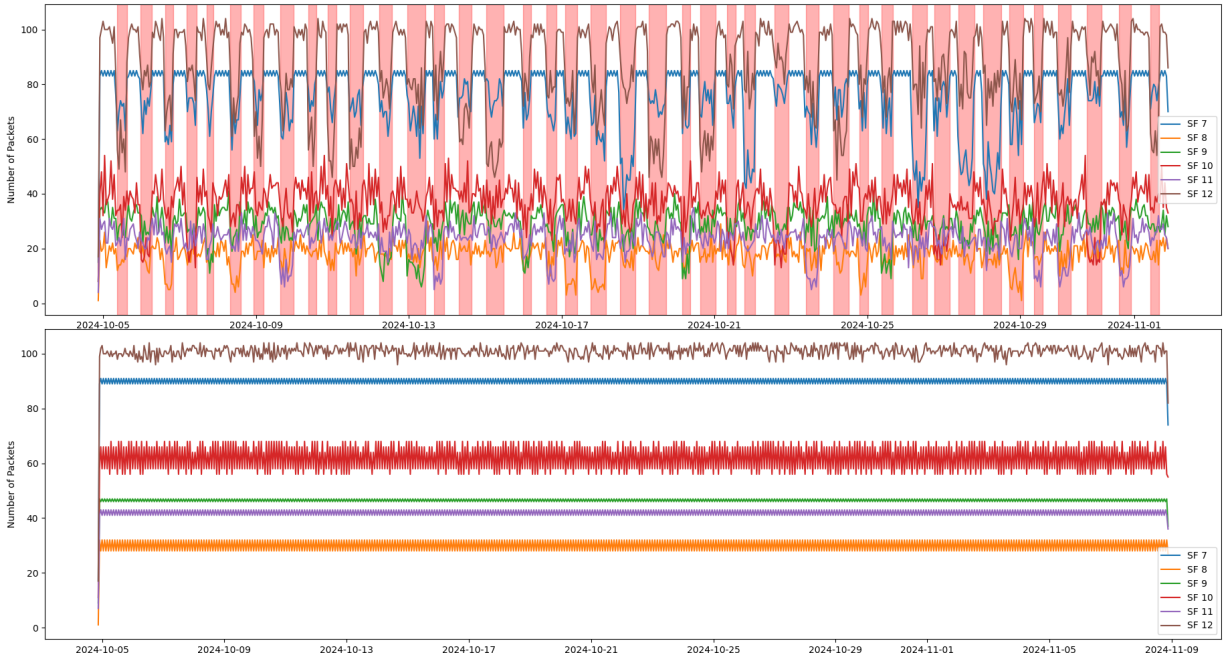


Figure 8.2: Comparison between network with normal traffic and jammed traffic

Jamming Duration and Impact – The jamming event spans the entire simulation, with random activation intervals to simulate unpredictable attacks. This provides insights into network resilience and recovery time after jamming events. This simulation evaluates LoRaWAN network resilience under various jamming attacks. The jammer's dynamic use of frequencies and SFs offers insights into the effectiveness of adaptive interference strategies. As an example, Fig. 8.2 shows traffic patterns for different SFs in both jammed (top) and normal (bottom) scenarios. The top plot shows packet transmission over time by SF, with red-shaded regions indicating jammer activity. Noticeable drops in packet count suggest the jammer's impact. The bottom plot instead shows consistent packet counts without jamming, highlighting even more the jammer's effect in the top plot.

Impact on LoRaWAN ADR – Together with the PDR, the ADR is also heavily impacted, which in turn have a great impact on both the overall network performance and the single ED. When a jamming attack substantially increases packet loss or degrades SNR, the LoRaWAN ADR algorithm reacts by raising the device's SF and/or transmit power to compensate for the perceived link weakness. In many implementations, this adjustment happens relatively quickly, because the ADR logic monitors consecutive missed acknowledgments, counting retransmissions too, or negative link margins to escalate SF (e.g., SF7 to SF8). A simplified form of the link margin calculation can be written as:

$$\text{LinkMargin} \;=\; \big(\text{SNR}_{\text{measured}} \;-\; \text{SNR}_{\text{required}}(\text{SF}_{\text{current}})\big) \;-\; \Delta_{\text{margin}},$$

where Δ_{margin} is an extra safety margin (e.g., 2–6 dB) which is implementation-dependent. If the link margin stays below zero for several uplinks, the ED (or the NS via downlink commands) typically increments SF (e.g., SF7 $\rightarrow$ SF8) or the transmit power in 2 dB steps until the link margin becomes acceptable.

Once the jammer is turned off, however, LoRaWAN's ADR tends to lower SF *much more*

cautiously, requiring a certain number of consecutive uplinks with improved quality parameters before deeming the link strong enough to justify a single downward SF adjustment. Let $\mathrm{SF}_{\mathrm{final}}$ be the high SF reached during jamming and $\mathrm{SF}_{\mathrm{initial}}$ the optimal (lower) SF. If $\mathrm{N}_{\mathrm{down}}$ is the number of successful transmissions needed for each downward SF step and P_{interval} is the time or number of uplinks between ADR evaluations, the time to revert can be approximated by:

$$T_{\mathrm{revert}} \;=\; \sum_{k=\mathrm{SF}_{\mathrm{final}}}^{\mathrm{SF}_{\mathrm{initial}}} \Big(\mathrm{N}_{\mathrm{down}} \times P_{\mathrm{interval}}\Big).$$

Because $\mathrm{N}_{\mathrm{down}}$ can be relatively large, a node forced to SF12 can remain there for dozens of transmissions, incurring substantial performance penalties.

From a network-wide perspective, high SF transmissions significantly increase the ToA. A simplified expression for LoRaWAN airtime calculation is:

$$T_{\mathrm{air}}(\mathrm{SF}) \;\approx\; \frac{2^{\mathrm{SF}}}{\mathrm{BW}} \times N_{\mathrm{syn}},$$

where N_{syn} is the number of symbols of the communication, indicating exponential growth in transmission duration with increasing SF. Longer transmissions increase the risk of collisions for all devices, reducing the overall PDR. For individual EDs, a prolonged stay at high SF also impacts energy consumption. Assuming a fixed transmission power P_{tx}, recalling that using higher transmission power indeed cause higher battery usage, $I_{\mathrm{tx}}(P_{tx})$ is the transmit current draw with said transmission power and V the supply voltage, the energy used per transmission is roughly:

$$E_{\mathrm{tx}} \;=\; I_{\mathrm{tx}}(P_{tx}) \times V \times T_{\mathrm{air}}(\mathrm{SF}),$$

so an extended T_{air} leads to higher battery drain. Consequently, even a single, long enough, jamming event can cause much longer aftermath consequences, keeping EDs at inflated SF and transmit power for a considerable period, thereby increasing their energy consumption and reducing network capacity until ADR slowly readjusts to optimal operating points. This underscores the importance of faster ADR recovery strategies or jamming detection mechanisms to mitigate prolonged performance degradation even after the jammer is deactivated.

8.5 Countermeasures

8.5.1 Detection

In this chapter, we examine two approaches for detecting jamming attacks in LoRaWAN networks: a statistical method and a machine learning-based technique. The statistical approach employs Z-score and IQR to detect anomalies by identifying deviations from expected network behavior. For the machine learning approach, advanced models such as Long Short-Term Memory (LSTM), RNN, and Gated Recurrent Unit (GRU) are used to capture temporal dependencies and identify jamming events based on network traffic patterns.

Statistical Approach

The statistical approach for jammer detection in LoRaWAN networks uses techniques like Z-score and IQR to identify deviations from normal network behavior, which may indicate jamming activity. These methods detect anomalies by quantifying deviations from expected packet transmission rates.

Z-score – The Z-score is a statistical measure indicating how many standard deviations a data point is from the mean of the distribution, defined as:

$$Z = \frac{x - \mu}{\sigma}$$

where:

- x: observed value (e.g., packet count during a given time interval)
- μ: mean of the data (e.g., average packet count over a period)
- σ: standard deviation of the data

In jammer detection, a high Z-score suggests a significant deviation from normal packet counts. For example, a sudden drop in packet count (resulting in a highly negative Z-score) could indicate jamming activity. The Z-score method is effective when traffic follows a stable distribution, but it may struggle with highly dynamic traffic due to threshold-setting challenges.

Interquartile Range (IQR) – The IQR is a measure of statistical dispersion, calculated as the difference between the third quartile (Q_3) and the first quartile (Q_1):

$$IQR = Q_3 - Q_1$$

$$\text{Anomaly if:} \quad x < Q_1 - 1.5 \times IQR \quad \text{or} \quad x > Q_3 + 1.5 \times IQR$$

The IQR method detects sudden decreases in packet counts that fall outside normal variability, indicating possible jamming. Its non-parametric nature makes it suitable for environments with highly variable traffic.

Machine Learning Approach

Statistical methods like Z-score require a predetermined observation window, and their effectiveness depends on this parameter. Alternatively, RNNs offer a more flexible approach by retaining information from previous time steps, making them effective for analyzing temporal data.

Recurrent Neural Networks – RNNs are artificial neural networks characterized by cyclically connected neurons, allowing them to retain information about past states. This makes RNNs particularly suited for processing sequential or temporal data, as they can model dynamic behaviors that depend on prior inputs.

Figure 8.3 shows the basic structure of an RNN, where each layer's output serves as input for the next layer, effectively creating internal memory. Figure 8.4 illustrates common types of RNNs: Vanilla RNN, LSTM, and GRU. Each line represents a full vector, yellow circles indicate pairwise operations, and yellow boxes represent neural network layers.

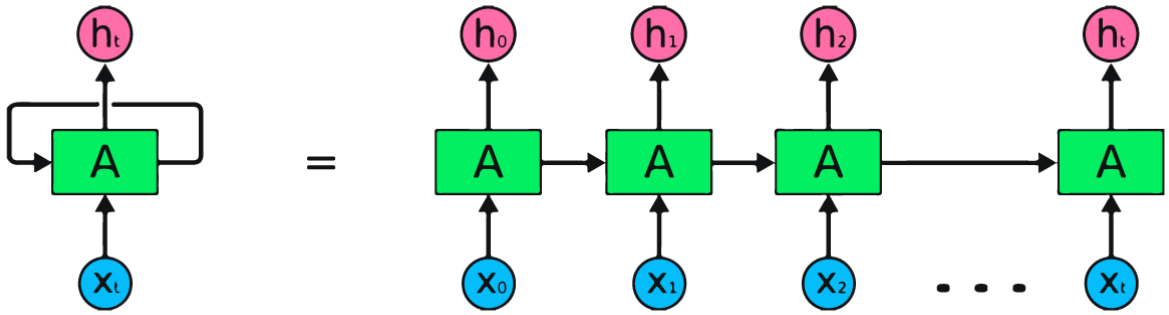


Figure 8.3: Top-level diagram of an RNN

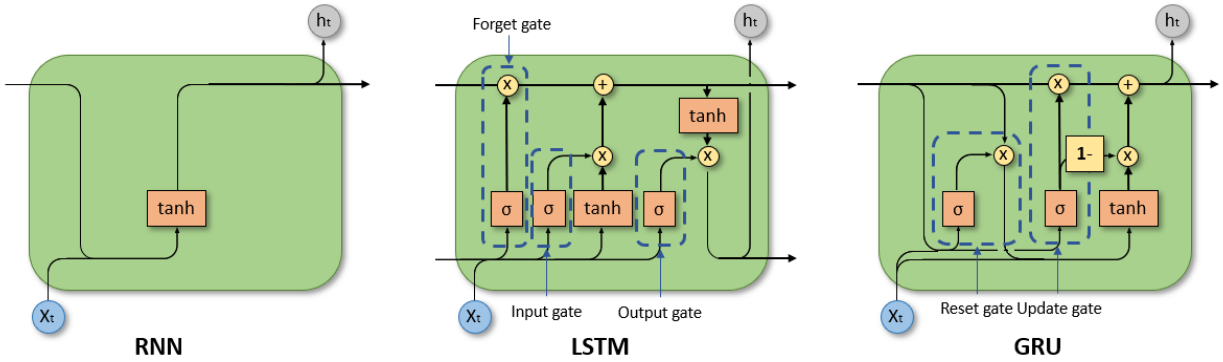


Figure 8.4: Different types of RNN: (a) Vanilla, (b) LSTM, (c) GRU (from [88])

Among the most widely used RNNs is the LSTM network, designed to capture long-term dependencies in sequential data [89]. The GRU is a variation of LSTM, introduced by Chung et al. [90], that combines the forget and input gates into a single update gate.

Dataset Preparation for Machine Learning Training

The dataset used for anomaly detection model development spanned 4 weeks, which provided a good balance between data richness and computational efficiency. The dataset, exported from the ns-3 simulator, consisted of 180,000 packets with end devices transmitting at regular 20-minute intervals. A channel-oblivious jammer was active intermittently, creating a realistic mix of normal and jammed traffic to improve the models' ability to distinguish between these scenarios. The raw dataset was grouped and resampled into 30-minute intervals. The data was normalized using a scaler to ensure consistent feature ranges, facilitating effective neural network learning.

Model Hyperparameters – Tab. 8.1 shows the best hyperparameters found using the Optuna framework [91] for RNN, LSTM, and GRU models, optimized for performance and generalization.

The key hyperparameters are described below:

- **Hidden Layer Size**: Number of neurons in each hidden layer, affecting model capacity and complexity.
- **Number of Layers**: Model depth, allowing for hierarchical learning but increasing computational cost.

Hyperparameter	RNN	LSTM	GRU
Hidden Layer Size	64	128	128
Number of Layers	1	2	2
Dropout	0.413	0.438	0.313
Learning Rate	0.0009373	0.0002931	0.0007962
Weight Decay	0.00489	0.00782	0.00588
Epochs	89	93	96
Momentum	0.804	0.923	0.9407
Alpha	0.829	0.960	0.9314
Patience	18	11	8
n_splits	9	9	4
Batch Size	32	32	32

Table 8.1: Best Hyperparameters for RNN, LSTM, and GRU models found using Optuna [91].

- **Dropout**: Regularization method that prevents overfitting by randomly deactivating neurons during training.
- **Learning Rate**: Step size for weight updates during optimization, controlling how fast the model learns.
- **Weight Decay**: L2 regularization term added to the loss function to reduce overfitting.
- **Epochs**: Number of times the model sees the entire dataset during training, balancing learning and overfitting risks.
- **Momentum**: Helps gradient descent optimization by smoothing weight updates using past gradients.
- **Alpha**: Controls the balance between L1 and L2 loss components in regularization, influencing error sensitivity.
- **Patience**: Early stopping criterion defining how many epochs to wait without validation improvement.
- **n_splits**: Number of folds in cross-validation for robust model evaluation.
- **Batch Size**: Number of samples processed before weight update; larger batch size gives more stable gradients.

Time-Series Cross-Validation – In time-series cross-validation, the data is split into sequential folds. The training process begins by using the earliest portion of the data, while the subsequent time periods are used for validation. As the validation progresses through each fold, the training set is expanded by adding more recent observations, and the validation set always remains ahead of the training data in time. This method mimics real-world forecasting or anomaly detection scenarios where future events are predicted based on past observations.

This approach ensures that the model is tested on unseen future data, maintaining the temporal causality and avoiding data leakage, which would occur if future information were incorporated into the training process.

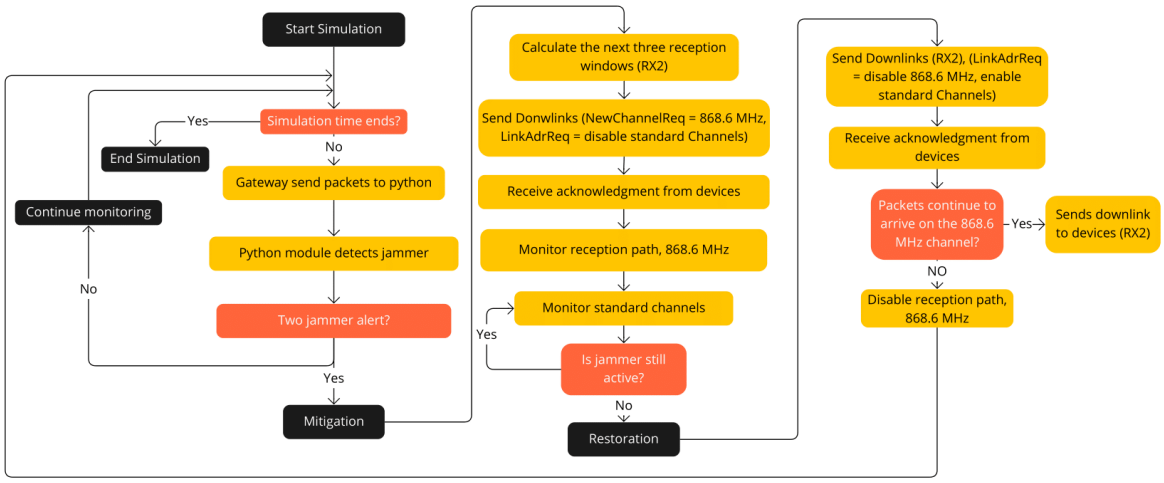


Figure 8.5: Network Mitigation and Recovery flow graph

8.5.2 Mitigation

The mitigation strategy, presented in Fig. 8.5 aims to minimize the impact of jamming attacks on LoRaWAN networks by dynamically adapting network parameters and rerouting traffic to maintain communication reliability. The mitigation process involves real-time detection of jamming activity, followed by the implementation of countermeasures to restore network performance and prevent further disruption. The mitigation system implemented in the ns-3 GW, integrated with a Python detection module, allows decentralized management of interference due to jamming attacks on the LoRaWAN network. This process involves real-time attack detection, sending control commands to LoRaWAN devices, and dynamic management of transmission channels to minimize interference effects.

Jammer Mitigation

When a new time-window closes, the GW sends a UDP message to the python module in order to let it analyze the new flow of traffic. Whenever it detects a jammer The Python module sends back a message to the GW. To prevent false positives due to temporary congestion, the GW waits for two consecutive jammer detection messages before initiating mitigation. During the simulation, the GW continuously monitors packets received from EDs, maintaining a history for each device, including address, data rate, and last packet timestamp. This helps in calculating appropriate downlink reception windows during mitigation.

Once mitigation is triggered, the GW calculates the downlink receive windows for each LoRaWAN device, leveraging Class A features where devices open receiving windows after uplink transmissions. The GW sends downlink packets to the EDs containing instructions to switch to a new frequency (868.6 MHz) and disable standard channels (868.1, 868.3, 868.5 MHz) to avoid jamming interference. The downlink packets are sent to the second receiving window of the EDs using 869.525 MHz with SF12, as per the LoRaWAN standard. By sending a single packet containing both MAC commands, the GW ensures that the requested changes are applied as soon as the ED acknowledges. During mitigation, one of the GW's available reception paths is dedicated

to receiving packets on the 868.6 MHz frequency, ensuring successful reception without interference for EDs that have received new instructions.

The GW continuously monitors the standard LoRa channels to detect when the jammer becomes inactive. A dynamic threshold is used to count packets, and once two consecutive detections confirm jammer deactivation, the "Restore Network" mechanism is activated. This mechanism sends a downlink to each device to disable the 868.6 MHz frequency and re-enable the standard LoRa channels.

If any devices do not receive the downlink correctly, the GW resends these messages once it starts receiving packets from those devices on the 868.6 MHz frequency, ensuring full restoration of the network. Once no packets are received on 868.6 MHz, the dedicated reception path is disabled, and the mitigation module resumes monitoring for new jamming interferences.

This decentralized mitigation algorithm allows each GW to independently manage its coverage area, making the LoRaWAN network more scalable and resilient to jamming attacks without relying on a central NS.

8.6 Evaluation and Results

8.6.1 Detection Evaluation Metrics

In this section, we evaluate the performance of both statistical and machine learning-based approaches for jammer detection in LoRaWAN networks. The evaluation metrics include accuracy, precision, recall, and F1-score, which are used to assess the models' effectiveness in identifying jamming activity.

Accuracy – Accuracy measures the fraction of correct predictions (both true positives and true negatives) over the total number of instances. It provides a general overview of the model's effectiveness in classification.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{8.5}$$

Precision – Precision calculates the proportion of true positive predictions among all positive predictions made by the model. It helps in evaluating the relevance of positive results.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{8.6}$$

Recall – Recall, also known as sensitivity, measures the proportion of true positives correctly identified by the model out of all actual positives. It indicates how well the model is detecting positive instances.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{8.7}$$

F1-Score – F1-score is the harmonic mean of precision and recall, providing a balance between the two. It is particularly useful when there is an uneven class distribution, ensuring both false positives and false negatives are accounted for.

$$\text{F1-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{8.8}$$

Model	Accuracy	Precision	Recall	F1-Score
Z-score	0.7991	0.8839	0.6793	0.7684
IQR	0.8296	0.9133	0.7204	0.8054
RNN	0.818	0.856	0.839	0.847
LSTM	0.846	0.931	0.806	0.864
GRU	0.843	0.909	0.823	0.864

Table 8.2: Summary of performance metrics for different detection models.

8.6.2 Detection Results

Table 8.2 summarizes the performance of the various detection techniques, including Z-score, IQR, RNN, LSTM, and GRU, on both training and test datasets. This allows for a quick comparison of the models' effectiveness.

High-Level Observations

Both Statistical models and Machine Learning models demostrate different levels of precision in detecting jamming attacks, highlighting their difference.

Statistical Models (Z-score, IQR): The statistical models demonstrated decent performance in terms of accuracy and precision, meaning they were fairly effective in correctly classifying whether jamming was present. However, their struggle with recall indicates that these models missed a higher number of actual jamming events, leading to a higher rate of false negatives. A low recall rate means that in real-world dynamic network conditions, these models might fail to detect some jamming events, especially when the jamming is subtle or variable. This unreliability in consistent detection makes them less suitable for applications where missing a jamming event could have serious consequences.

The statistical methods are relatively simple to implement and require fewer computational resources compared to machine learning models. As a result, they are well-suited for environments where computational capacity is limited, or for scenarios requiring lightweight detection without stringent accuracy demands.

Machine Learning Models (RNN, LSTM, GRU): The machine learning models generally outperformed the statistical approaches across all metrics, particularly excelling in recall, meaning they were more capable of identifying jamming activity whenever it was present.

LSTM and GRU models, in particular, achieved the highest performance metrics. Their ability to maintain higher accuracy on test data suggests that these models generalize well to unseen data, making them reliable for deployment in production environments where network conditions may change over time.

RNN models also showed good results but were slightly less effective than LSTM and GRU in terms of recall. This suggests that while RNNs are able to model temporal dependencies, they might miss certain jamming events due to their limited ability to capture long-term dependencies compared to LSTMs and GRUs.

Machine learning models are more computationally intensive, requiring sufficient processing power and memory. Additionally, they need proper hyperparameter tuning to avoid issues like overfitting, where the model becomes too closely fitted to the training data and performs poorly on new, unseen data.

Key Machine Learning Model Insights

LSTM and GRU Models – The LSTM and GRU models both achieved F1-scores of 0.864, making them the best choices for scenarios requiring a high degree of detection accuracy. Their ability to capture both short-term and long-term dependencies in the data means they are effective at distinguishing between normal traffic patterns and those disrupted by jamming, even when the patterns are subtle. These models are especially useful in critical network environments (e.g., industrial IoT or smart cities) where missing a jamming event could lead to safety or operational issues. However, their computational requirements mean they are better suited for scenarios where sufficient hardware is available, such as edge devices or cloud-based systems.

RNN Model – The RNN model also demonstrated solid performance, although it was slightly behind LSTM and GRU in recall. This suggests that it might miss certain jamming events, especially those that require more complex temporal tracking. However, it offers a good compromise between computational requirements and performance and may be suitable for applications where a slight decrease in recall is acceptable in exchange for reduced computational demands.

Based on the collected information, the choice of detection model depends on the specific application requirements:

- For environments where high accuracy, precision, and recall are essential, and computational resources are available, LSTM and GRU models are the most suitable
- For applications requiring moderate detection capabilities with fewer computational demands, the RNN model provides a good balance
- For lightweight and quick detection with minimal computational requirements, statistical models like Z-score and IQR are viable options, keeping in mind the trade-off in detection completeness

8.6.3 Attack and Mitigation Results

The analysis of jamming attacks and the corresponding mitigation strategies provides insight into the effectiveness of the jammer and the ability of the network to recover under different conditions. This evaluation considers key network performance metrics like PDR, throughput, IAT, and jitter, to assess both the jamming efficiency and the mitigation's impact.

Jammer Efficiency

The jammer's efficiency is evident from its significant negative impact on all network performance metrics, as shown in Fig. 8.6 when no mitigation is applied:

- **PDR Over Time:** Without mitigation, the PDR fluctuates significantly due to jammer interference, often dropping below 50%. This illustrates the jammer's effectiveness in disrupting packet transmissions and causing considerable packet loss.
- **Throughput Over Time:** During jamming periods, throughput drops considerably, ranging between 10 to 15 kbps. This demonstrates the jammer's ability to severely limit data transmission rates, reducing overall network efficiency.

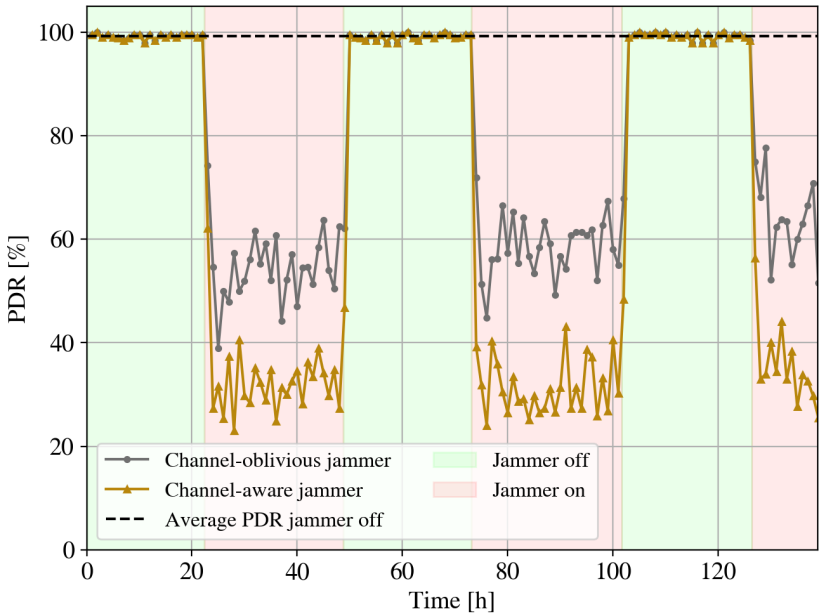


Figure 8.6: Packet Delivery Ratio (PDR) comparing the impact of channel-obliviuos jammer, channel-aware jammer and without external interference.

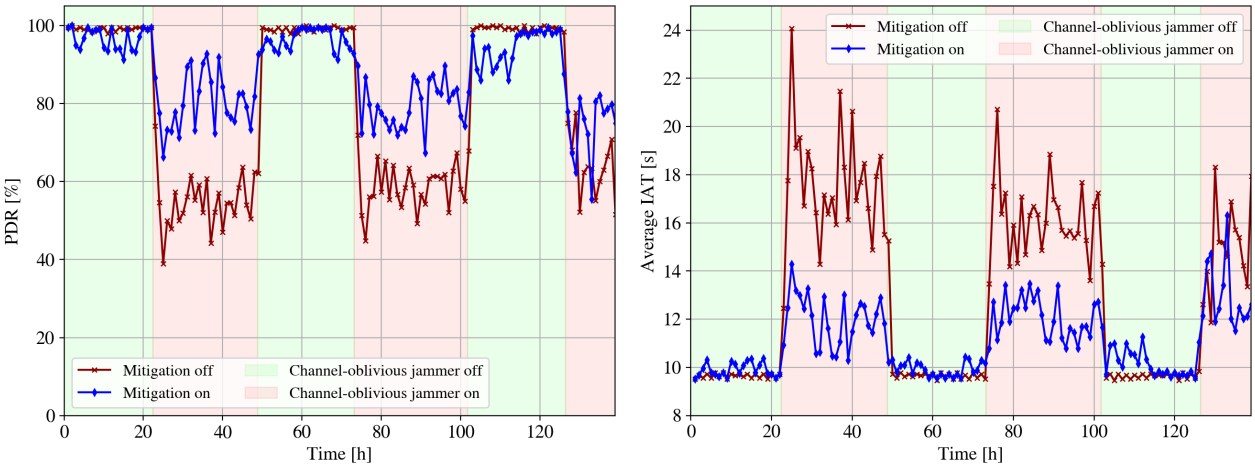
- **Average IAT Over Time:** Without mitigation, the average IAT increases significantly, reaching values as high as 24 seconds. This indicates delayed packet delivery and highlights the jammer's impact on increasing transmission delays.
- **Jitter Over Time:** Jitter is highly variable without mitigation, often exceeding 25-30 ms, which suggests unstable packet timing and significant network disruption caused by the jammer.

Indeed the jammer is highly effective in reducing network performance, causing decreased PDR and throughput, along with increased IAT and jitter.

Mitigation Efficiency

The effectiveness of the mitigation mechanism was evaluated against both channel-oblivious and channel-aware jammers, revealing varying levels of success:

- **Channel-Oblivious Jammer:** The mitigation strategy had a clear positive effect on network performance metrics:
  - **PDR:** With mitigation, presented in Fig. 8.7a, PDR remains more stable and higher, generally around 80%, indicating that mitigation measures help maintain a higher number of successful packet transmissions. The mitigation improved PDR by approximately 30-50% compared to scenarios without mitigation.
  - **Throughput:** Throughput is maintained at a higher and more consistent level, typically between 20 to 27 kbps, showing up to a 50% improvement compared to scenarios without mitigation. This reflects the mitigation's success in maintaining data flow despite jamming.



(a) Comparison of Packet Delivery Ratio (PDR) with and without mitigation during channel-oblivious jamming.

(b) Average IAT analysis with and without mitigation during channel-oblivious jamming.

Figure 8.7: Results of mitigation strategies during channel-oblivious jamming.

  - **IAT:** The average IAT is reduced and stays below 15 seconds with mitigation, demonstrating a reduction in packet delays and more regular packet intervals, with an improvement of around 30-40%, shown in Fig. 8.7b.
  - **Jitter:** Jitter values are reduced to below 15 ms, indicating improved timing stability. The reduction in jitter highlights the mitigation's effectiveness in achieving smoother packet delivery, reducing variability by about 40-50%.

- **Channel-Aware Jammer:** The mitigation strategy provided only slight improvements against the channel-aware jammer:
  - **PSR and Throughput:** Both metrics showed only marginal improvements, with overall performance remaining well below acceptable thresholds. The channel-aware jammer's adaptive behavior effectively countered typical mitigation attempts.
  - **IAT and Jitter:** Both metrics exhibited high variability, indicating continued instability and the network's inability to stabilize. The channel-aware jammer's sophistication rendered the mitigation largely ineffective.

8.7 Conclusion and Future Work

In this chapter, we introduced a range of countermeasures designed to mitigate the effects of jamming attacks on LoRaWAN networks. These countermeasures were categorized into detection techniques, proactive strategies, and reactive mechanisms. We examined two distinct detection methods: a statistical approach utilizing Z-score and IQR, and a machine learning-based approach incorporating models like LSTM, RNN, and GRU. Each approach was evaluated based on its accuracy, precision, recall, and overall effectiveness in real-time scenarios.

The mitigation mechanism, implemented in the ns-3 simulator and integrated with the detection models, demonstrated significant improvements in network performance when dealing with channel-oblivious jammers. Metrics such as PDR, throughput, jitter, and IAT showed considerable recovery

when mitigation measures were in place. By dynamically managing reception paths and transmission parameters, the GW was able to minimize the jammer's impact, thus maintaining network stability during attacks.

However, when confronted with a more sophisticated channel-aware jammer, the mitigation strategy's effectiveness diminished significantly. The adaptive nature of the channel-aware jammer made it challenging for the network to counteract its effects, as it could selectively target active communication channels. This highlighted the need for more advanced mitigation strategies to deal with adaptive jamming attacks effectively.

The "Restore Network" mechanism added an additional layer of resilience by allowing the network to revert to its normal operating state once the jamming ceased. This ensured that the network could recover promptly, returning to standard LoRaWAN channels and minimizing the long-term impact of the jamming attack.

Additionally, the detection mechanism developed for identifying jamming attacks has shown potential for detecting other forms of interference as well, such as inter-network interferences. This capability makes it a valuable component of a more comprehensive smart ADR algorithm aimed at enhancing network resilience. By incorporating interference detection, the system can make more informed decisions to adjust network parameters dynamically, thereby improving overall performance and robustness.

Chapter 9

Conclusions

In this final chapter, we bring together the comprehensive body of work presented in this thesis, which focused on enhancing the resilience and efficiency of Low-Power Wide-Area Networks (LPWANs), particularly in the context of Internet of Things (IoT) networks and edge computing. The core aspects discussed include privacy vulnerabilities in Bluetooth Low Energy (BLE) networks, decentralized LoRaWAN architecture for improved scalability and reliability, and countermeasures for mitigating jamming attacks and interferences in LoRaWAN. Together, these contributions aim to provide a deeper understanding of the challenges and advancements in LPWANs and IoT network management, while also laying the groundwork for future developments.

The first part of the thesis focused on the BLENDER system, which exposed the limitations of Medium Access Control (MAC) address randomization as a privacy measure in BLE networks. BLE, a cornerstone technology for IoT devices, aims to ensure privacy through the use of random MAC addresses. However, through a combination of passive and active scanning methods, BLENDER revealed how such randomization could be bypassed, exposing BLE devices to tracking risks. This demonstrated that relying solely on MAC address randomization is insufficient for privacy protection, highlighting the need for more advanced privacy-preserving techniques in IoT networks. The findings underscore the necessity of integrating more sophisticated security and privacy measures in BLE-enabled IoT environments, where edge devices often operate with limited security capabilities.

The second major contribution was the development and evaluation of DeLoRaN, a decentralized architecture for LoRaWAN networks that leverages blockchain technology to enhance scalability, resilience, and security. LPWANs like LoRaWAN are widely used in IoT due to their low power consumption and wide coverage, making them suitable for connecting a massive number of End Devicess (EDs). However, traditional centralized systems such as Chirpstack face issues related to scalability, Single Point of Failure (SPoF), and vulnerability to attacks. DeLoRaN addressed these challenges by decentralizing control across multiple Network Controllers (NCs) and utilizing blockchain for secure, distributed management. The experimental evaluation demonstrated that DeLoRaN efficiently distributed control, handled up to 30,000 devices, and maintained low response times without performance degradation. This contribution is particularly relevant for the future of IoT networks, as it offers a scalable solution capable of adapting to the increasing demands of large-scale deployments, while also benefiting from edge computing principles to improve resilience and minimize latency.

The final component of the thesis dealt with the development of countermeasures for jamming

attacks and interferences in LoRaWAN networks. Given the susceptibility of wireless communication channels to interference, jamming represents a significant threat to LPWANs, which are designed to support long-range communication with limited power. The mitigation strategies were designed to address both simple and advanced jamming attacks, employing both statistical and machine learning-based detection techniques. The integration of these countermeasures into the ns-3 simulator demonstrated significant improvements in network performance, particularly against channel-oblivious jammers. However, the limitations against more sophisticated channel-aware jammers highlighted the ongoing need for adaptive and robust solutions. Importantly, the detection mechanism also showed promise in identifying other forms of interference, paving the way for its integration into a smart Adaptive Data Rate (ADR) algorithm. This holistic approach could significantly improve network resilience by dynamically adjusting parameters to optimize performance in the presence of interference, effectively making LoRaWAN networks more adaptable and robust.

In conclusion, this thesis provides a comprehensive exploration of critical issues affecting the resilience, privacy, and scalability of LPWANs, particularly focusing on IoT networks and edge computing environments. The work underscores the need for a multi-faceted approach in addressing the challenges faced by modern IoT networks—from enhancing privacy in BLE devices and implementing decentralized control in LoRaWAN to developing robust countermeasures against jamming attacks. Each of these contributions plays a vital role in advancing the state-of-the-art in IoT and LPWAN management, offering practical solutions to improve the efficiency and security of edge devices and networks. As IoT deployments continue to grow, these advancements lay the foundation for building more resilient, scalable, and secure network infrastructures that are essential for the next generation of smart environments.

Bibliography

[1] Kevin Ashton. That 'internet of things' thing. *RFID Journal*, 22(7):97–114, 2009.

[2] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7):1645–1660, 2013.

[3] IEEE. Ieee standard for local and metropolitan area networks—part 15.4: Low-rate wireless personal area networks (lr-wpans), 2003.

[4] Bluetooth SIG Alliance. Bluetooth core specification v.4.0. `https://www.bluetooth.com/specifications/specs/core-specification-4-0/`, 2019. [Online; accessed January 17, 2025].

[5] Ayoub M Kamal, Mohammad Mosarof Alam, Abdullah Al Baki Sajak, and Mohd Su'ud. Requirements, deployments, and challenges of lora technology: A survey. *Computational Intelligence and Neuroscience*, 2023:5183062, 2023.

[6] LoRa Alliance. Lorawan 1.0.3 specification. `https://lora-alliance.org/resource_hub/lorawan-specification-v1-0-3/`.

[7] PJ Escamilla-Ambrosio and A. Rodríguez-Mota. Distributing computing in the internet of things: cloud, fog and edge computing overview. In *NEO 2016: Results of the International Conference on NEO*, pages 55–66. Springer, 2018.

[8] Geoff Mulligan. The 6lowpan architecture. In *Proceedings of the 4th Workshop on Embedded Networked Sensors*, EmNets '07, page 78–82, New York, NY, USA, 2007. Association for Computing Machinery.

[9] M. Fazio, R. Ranjan, M. Girolami, and J. Taheri. A note on the convergence of iot, edge, and cloud computing in smart cities. *IEEE International Conference on Cloud Computing*, 2018.

[10] S. Kanungo. Edge-to-cloud intelligence: Enhancing iot devices with machine learning and cloud computing. *International Peer-Reviewed Journal*, 2019.

[11] Verma Rajat, Kumar Mishra Prashant, Nagar Vishal, and Mahapatra Satyasundara. *Internet of Things in Smart Homes: A Review*, chapter 9. Taylor & Francis, 2021.

[12] Rishabh Jain, Rakhi Kumari, and Sudeep Tanwar. A survey on iot security: Challenges and solutions. *Computers & Security*, 119:100429, 2022.

[13] Michael Lee. A survey on the real-world cyberattacks on the industrial internet of things. *TechRxiv*, June 07 2023.

[14] C. Bisdikian. An overview of the bluetooth wireless technology. *IEEE Communications Magazine*, 39:86–94, 2001.

[15] Bluetooth Special Interest Group (SIG). Bluetooth core specification 5.0. `https://www.bluetooth.com/specifications/bluetooth-core-specification/`, 2016. Available online at `https://www.bluetooth.com/specifications/bluetooth-core-specification/`.

[16] Bluetooth SIG Alliance. Mesh profile specification v.1.0.1. `https://www.bluetooth.com/specifications/specs/mesh-profile-1-0-1/`, 2019. [Online; accessed January 17, 2025].

[17] Ángel Hernández-Solana, David Pérez-Díaz-De-Cerio, Mario García-Lozano, Antoni Vives Bardají, and José Valenzuela. Bluetooth mesh analysis, issues, and challenges. *IEEE Access*, 8:53784–53800, 2020.

[18] Rashmi Sharan Sinha, Yiqiao Wei, and Seung-Hoon Hwang. A survey on lpwa technology: Lora and nb-iot. *ICT Express*, 3(1):14–21, 2017.

[19] J. Haxhibeqiri, E. De Poorter, I. Moerman, and J. Hoebeke. A survey of lorawan for iot: From technology to application. *Sensors (Basel)*, 18(11):3995, 2018.

[20] J. de Carvalho Silva, J. J. P. C. Rodrigues, A. M. Alberti, P. Solic, and A. L. L. Aquino. Lorawan – a low power wan protocol for internet of things: A review and opportunities. In *2017 2nd International Multidisciplinary Conference on Computer and Energy Science (SpliTech)*, pages 1–6, 2017.

[21] LoRa Alliance. Lorawan 1.1 specification. `https://lora-alliance.org/wp-content/uploads/2020/11/lorawantm_specification_-v1.1.pdf`.

[22] J. Han and J. Wang. An enhanced key management scheme for lorawan. *Cryptography*, 2(4):34, 2018.

[23] Emekcan Aras, Gowri Ramachandran, Piers Lawrence, and Danny Hughes. Exploring the security vulnerabilities of lora. In *2017 IEEE Cyber Science and Technology Congress (CYBConf)*, pages 1–6, 2017.

[24] X. Yang, E. Karampatzakis, C. Doerr, and F. Kuipers. Security vulnerabilities in lorawan. In *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 129–140, 2018.

[25] G. Bernardinetti, F. Mancini, and G. Bianchi. Disconnection attacks against lorawan 1.0.x abp devices. In *2020 Mediterranean Communication and Computer Networking Conference (MedComNet)*, 2020.

[26] T. C. M. Dönmez and E. Nigussie. Security of lorawan v1.1 in backward compatibility scenarios. In *Procedia Computer Science*, volume 134, pages 51–58. Elsevier, 2018.

[27] I. Butun, N. Pereira, and M. Gidlund. Demystifying security of lorawan v1.1. *MDPI AG*, 2018.

[28] Pietro Spadaccino, Domenico Garlisi, Francesca Cuomo, Giorgio Pillon, and Patrizio Pisani. Discovery privacy threats via device de-anonymization in lorawan. In *2021 Mediterranean Communication and Computer Networking Conference (MedComNet)*, 2021.

[29] Arup Barua, Md Abdullah Al Alamin, Md. Shohrab Hossain, and Ekram Hossain. Security and privacy threats for bluetooth low energy in iot and wearable devices: A comprehensive survey. *IEEE Open Journal of the Communications Society*, 3:251–281, 2022.

[30] Meredydd Williams, Jason R. C. Nurse, and Sadie Creese. The perfect storm: The privacy paradox and the internet-of-things. *2016 11th International Conference on Availability, Reliability and Security (ARES)*, pages 644–652, 2016.

[31] Jeremy Martin, Douglas Alpuche, Kristina Bodeman, Lamont Brown, Ellis Fenske, Lucas Foppe, Travis Mayberry, Erik C. Rye, Brandon Sipes, and Sam Teplov. Handoff all your privacy – a review of apple's bluetooth low energy continuity protocol. *Proceedings on Privacy Enhancing Technologies*, 2019:34 – 53, 2019.

[32] Angeliki Aktypi, Jason R. C. Nurse, and Michael Goldsmith. Unwinding ariadne's identity thread: Privacy risks with fitness trackers and online social networks. *Proceedings of the 2017 on Multimedia Privacy and Security*, 2017.

[33] Ellis Fenske, Dane Brown, Jeremy Martin, Travis Mayberry, Peter Ryan, and Erik Rye. Three years later: A study of mac address randomization in mobile devices and when it succeeds. *Proceedings on Privacy Enhancing Technologies*, 2021:164–181, 07 2021.

[34] Tadayoshi Kohno, Andre Broido, and Kimberly C. Claffy. Remote physical device fingerprinting. *2005 IEEE Symposium on Security and Privacy (S&P'05)*, pages 211–225, 2005.

[35] Massimo Perri, Francesca Cuomo, and Pierluigi Locatelli. Blender - bluetooth low energy discovery and fingerprinting in iot. In *2022 20th Mediterranean Communication and Computer Networking Conference (MedComNet)*, pages 182–189, 2022.

[36] Pierluigi Locatelli, Massimo Perri, Daniel Mauricio Jimenez Gutierrez, Andrea Lacava, and Francesca Cuomo. Device discovery and tracing in the bluetooth low energy domain. *Computer Communications*, 202:42–56, 2023.

[37] Microchip Technology. Bluetooth® Low Energy Packet Types. `https://microchipdeveloper.com/wireless:ble-link-layer-packet-types`, 2020. Accessed: 2022-04-07.

[38] Guillaume Celosia and Mathieu Cunche. Fingerprinting bluetooth-low-energy devices based on the generic attribute profile. In *Proceedings of the 2nd international ACM workshop on security and privacy for the internet-of-things*, pages 24–31. ACM, 2019.

[39] Talla Issoufaly and Pierre-Ugo Tournoux. Bleb: Bluetooth low energy botnet for large scale individual tracking. In *2017 1st International Conference On Next Generation Computing Applications (NextComp)*, pages 115–120. IEEE, 2017.

[40] Aveek K Das, Parth H Pathak, Chen-Nee Chuah, and Prasant Mohapatra. Uncovering privacy leakage in ble network traffic of wearable fitness trackers. In *Proceedings of the 17th international workshop on mobile computing systems and applications*, pages 99–104. ACM, 2016.

[41] Johannes K Becker, David Li, and David Starobinski. Tracking anonymized bluetooth devices. *Proceedings on Privacy Enhancing Technologies*, 2019(3):50–65, 2019.

[42] Guillaume Celosia and Mathieu Cunche. Saving private addresses: An analysis of privacy issues in the bluetooth-low-energy advertising mechanism. In *Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pages 444–453. ACM, 2019.

[43] Bluetooth SIG Alliance. Bluetooth core specification v.4.2. `https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=441541`, 2019. [Online; accessed January 17, 2025].

[44] Zaid Ahmad, S. J. Hashim, and F. Z. Rokhani. Lpwan state of the art: Trends and future directions. In *Proceedings of Smart City Innovations*, 2021.

[45] Yonghua Song, Jianhua Lin, Ming Tang, and Shenghui Dong. An internet of energy things based on wireless lpwan. *Engineering*, 3(4):460–466, 2017.

[46] Sigfox - https://www.sigfox.com/.

[47] Collins B. Mwakwata, Haris Malik, Muhammad Mahbub Alam, Yannick Le Moullec, Shahriar Parand, and Shahid Mumtaz. Narrowband internet of things (nb-iot): From physical (phy) and media access control (mac) layers perspectives. *Sensors (Basel)*, 19(11):2613, 2019.

[48] Gaurav Pathak, Jairo Gutierrez, and Saeed Ur Rehman. Security in low powered wide area networks: Opportunities for software defined network-supported solutions. *Electronics*, 9(8), 2020.

[49] Vamshi Sunku Mohan, Sriram Sankaran, Priyadarsi Nanda, and Krishnashree Achuthan. Enabling secure lightweight mobile narrowband internet of things (nb-iot) applications using blockchain. *Journal of Network and Computer Applications*, 219:103723, 2023.

[50] Mohammed Bouzidi, Ahmed Amro, Yaser Dalveren, Faouzi Alaya Cheikh, and Mohammad Derawi. Lpwan cyber security risk analysis: Building a secure iqrf solution. *Sensors*, 23(4), 2023.

[51] Djamel Eddine Kouicem, Abdelmadjid Bouabdallah, and Hicham Lakhlef. Internet of things security: A top-down survey. *Computer Networks*, 141:199–221, 2018.

[52] Kuburat Oyeranti Adefemi Alimi, Khmaies Ouahada, Adnan M. Abu-Mahfouz, and Suvendi Rimer. A survey on the security of low power wide area networks: Threats, challenges, and potential solutions. *Sensors*, 20(20), 2020.

[53] Yulei Wu, Zheng Yan, Kim-Kwang Raymond Choo, and Laurence T. Yang. Ieee access special section editorial: Internet-of-things big data trust management. *IEEE Access*, 7:65223–65227, 2019.

[54] Mehdi Bezahaf, Gaëtan Cathelain, and Tony Ducrocq. Bcwan: A federated low-power wan for the internet of things (industry track). In *Proceedings of the 19th International Middleware Conference Industry*, Middleware '18, page 54–60, New York, NY, USA, 2018. Association for Computing Machinery.

[55] Arnaud Durand, Pascal Gremaud, and Jacques Pasquier. Decentralized lpwan infrastructure using blockchain and digital signatures. *Concurrency and Computation: Practice and Experience*, 2020.

[56] Victor Ribeiro, Raimir Holanda Filho, and Alex Ramos. A secure and fault-tolerant architecture for lorawan based on blockchain. In *2019 3rd Cyber Security in Networking Conference (CSNet)*, pages 35–41, 2019.

[57] Mingxi Tan, Dazhi Sun, and Xiaohong Li. A secure and efficient blockchain-based key management scheme for lorawan. In *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–7, 2021.

[58] Jie Lin, Zengxiang Shen, Chunyan Miao, and Sha Liu. Using blockchain to build trusted lorawan sharing server. *International Journal of Crowd Science*, 1(3):270–280, 2017.

[59] Lujia Hou, Kai Zheng, Ziyue Liu, Xueyan Xu, and Tong Wu. Design and prototype implementation of a blockchain-enabled lora system with edge computing. *IEEE Internet of Things Journal*, 2020.

[60] Yang Wei, Kim Fung Tsang, and Hao Wang. An efficient and secure dag-based lorawan system. In *2023 IEEE 32nd International Symposium on Industrial Electronics (ISIE)*, pages 1–5, 2023.

[61] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[62] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, July 1982.

[63] Pierluigi Locatelli and Francesca Cuomo. Deloran: Decentralize lorawan network server through blockchain. In *2024 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, 2024.

[64] Konstantin Mikhaylov, Juha Petäjäjärvi, and Janne Janhunen. On lorawan scalability: Empirical evaluation of susceptibility to inter-network interference. In *2017 European Conference on Networks and Communications (EuCNC)*, pages 1–6, 2017.

[65] Karl Wüst and Arthur Gervais. Do you need a blockchain? In *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pages 45–54, 2018.

[66] Benchmark Hyperledger Fabric: https://www.hyperledger.org/blog/2023/02/16/benchmarking-hyperledger-fabric-2-5-performance.

[67] Pierluigi Locatelli, Pietro Spadaccino, and Francesca Cuomo. Hijacking downlink path selection in lorawan. In *2021 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2021.

[68] Lemogue Arnol, Ivan Martinez, Laurent Toutain, and Ahmed Bouabdallah. Federated iot roaming using private dns resolutions. 04 2022.

[69] Lu Hou, Kan Zheng, Zhiming Liu, Xiaojun Xu, and Tao Wu. Design and prototype implementation of a blockchain-enabled lora system with edge computing. *IEEE Internet of Things Journal*, 8(4):2419–2430, 2021.

[70] Laksh Bhatia, Michael Breza, Ramona Marfievici, and Julie A. McCann. Loed: The lorawan at the edge dataset. In *Proceedings of the Third Workshop on Data Acquisition To Analysis (DATA '20)*, Virtual Event, Japan, 2020. ACM.

[71] Pierluigi Locatelli, Pietro Spadaccino, and Francesca Cuomo. Realistic traffic modeling and performance evaluation of a blockchain-enabled lorawan. In *Accepted at 2025 IEEE Wireless Communications and Networking Conference (WCNC)*, 2025.

[72] Norhane Benkahla, Hajer Tounsi, Ye-Qiong Song, and Mounir Frikha. Review and experimental evaluation of adr enhancements for lorawan networks. *Telecommunication Systems*, 77, 05 2021.

[73] Luz E. Marquez, Alfonso Osorio, Maria Calle, Juan C. Velez, Antonio Serrano, and John E. Candelo-Becerra. On the use of lorawan in smart cities: A study with blocking interference. *IEEE Internet of Things Journal*, 7(4):2806–2815, 2020.

[74] Ivan Marino Martinez Bolivar. *Jamming on LoRaWAN Networks : from modelling to detection.* Theses, Institut National des Sciences Appliquées de Rennes, January 2021.

[75] Lluís Casals, Carles Gomez, and Rafael Vidal. The sf12 well in lorawan: Problem and end-device-based solutions. *Sensors*, 21(19), 2021.

[76] Stefano Di Pinto, Pierluigi Locatelli, Pietro Spadaccino, and Francesca Cuomo. Detection and mitigation of jamming attacks in lorawan using machine learning. In *Accepted at 2025 IEEE Wireless Communications and Networking Conference (WCNC)*, 2025.

[77] S. Li, U. Raza, and A. Khan. How agile is the adaptive data rate mechanism of lorawan? In *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018. [Online]. Available: `https://doi.org/10.1109/glocom.2018.8647469`.

[78] W. Xu, K. Ma, W. Trappe, and Y. Zhang. Jamming sensor networks: attack and defense strategies. *IEEE Network*, 2006. [Online]. Available: `https://doi.org/10.1109/mnet.2006.1637931`.

[79] D. Magrin, M. Capuzzo, and A. Zanella. A thorough study of lorawan performance under different parameter settings. *IEEE Internet of Things Journal*, jan 2020. [Online].

[80] C.-Y. Huang, C.-W. Lin, R.-G. Cheng, S. J. Yang, and S.-T. Sheu. Experimental evaluation of jamming threat in lorawan. In *Proceedings of the 2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, Kuala Lumpur, Malaysia, April 28-May 1 2019.

[81] I. Martinez, P. Tanguy, and F. Nouvel. On the performance evaluation of lorawan under jamming. In *Proceedings of the 2019 12th IFIP Wireless and Mobile Networking Conference (WMNC)*, Paris, France, September 11-13 2019.

[82] H. Ruotsalainen, G. Shen, J. Zhang, and R. Fujdiak. Lorawan physical layer-based attacks and countermeasures, a review. *Sensors*, 22(9):Article no. 3127, April 2022.

[83] M. Ingham, J. Marchang, and D. Bhowmik. Iot security vulnerabilities and predictive signal jamming attack analysis in lorawan. *IET Information Security*, 22(9), July 2020.

[84] D. Magrin, M. Centenaro, and L. Vangelista. Performance evaluation of lora networks in a smart city scenario. In *Proceedings of the 2017 IEEE International Conference on Communications (ICC)*, pages 1–7, Paris, France, 2017.

[85] A. Mpitziopoulos, D. Gavalas, C. Konstantopoulos, and G. Pantziou. A survey on jamming attacks and countermeasures in wsns. *IEEE Communications Surveys & Tutorials*, 2009. [Online]. Available: `https://doi.org/10.1109/surv.2009.090404`.

[86] Sharon J. Lagat. Detecting denial of service attacks in lorawan: Experimental analysis with machine learning predictive modelling to detect jamming and collision attacks on lora networks. *ResearchGate*, 2022. [Online]. Available: `https://doi.org/10.48550/arXiv.2202.00349`.

[87] Shawn G. Richardson. *LoRaWAN Sensor Network Jamming Detection and Mitigation Using Machine Learning in the Cloud*. PhD thesis, ProQuest, 2023. [Online]. Available: `https://doi.org/10.1109/JRS.2023.001`.

[88] C. Olah. Understanding lstm networks, 2016. [Online]. Available: `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`.

[89] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, Nov. 1997. [Online]. Available: `https://doi.org/10.1162/neco.1997.9.8.1735`.

[90] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *ArXiv*, 2014.

[91] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2623–2631, 2019.