Pontifícia Universidade Católica
do Rio de Janeiro

Dayrene Frómeta Fonseca

Machine Learning-Based MAC Protocols for LoRa IoT Networks

Dissertação de Mestrado

Thesis presented to the Programa de Pós–graduação em Engenharía Elétrica da PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Engenharía Elétrica.

Advisor : Prof. Rodrigo C. de Lamare
Co-advisor: Dr. Ewerton Longoni Madruga

Rio de Janeiro
February 2020

Pontifícia Universidade Católica
do Rio de Janeiro

Dayrene Frómeta Fonseca

Machine Learning-Based MAC Protocols for LoRa IoT Networks

Thesis presented to the Programa de Pós–graduação em Engenharía Elétrica da PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Engenharía Elétrica. Approved by the undersigned Examination Committee.

Prof. Rodrigo C. de Lamare
Advisor
CETUC – PUC-Rio

Dr. Ewerton Longoni Madruga
Co-advisor
Inmetro – Inmetro

Prof. Lukas Tobias Nepomuk Landau
CETUC – PUC-Rio

Prof. Cíntia Borges Margi
University of São Paulo – USP

Prof. Marco Antonio Grivet Mattoso Maia
CETUC – PUC-Rio

Rio de Janeiro, February the 15th, 2020

Dayrene Frómeta Fonseca

Rceived the Electronics and Telecommunication Engineering degree from Technological University of Havana José Antonio Echeverría (Havana, Cuba) in 2015.

Bibliographic data

Frómeta Fonseca, Dayrene

Machine Learning-Based MAC Protocols for LoRa IoT Networks / Dayrene Frómeta Fonseca; advisor: Rodrigo C. de Lamare; co-advisor: Ewerton Longoni Madruga . – Rio de Janeiro: PUC-Rio, Departamento de Engenharía Elétrica, 2020.

v., 126 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharía Elétrica.

Inclui bibliografia

1. Engenharia Elétrica – Teses. 2. Redes de Longa Distância de Baixa Potência;. 3. Protocolos de Controle de Acesso ao Médio;. 4. Modulação LoRa;. 5. LoRaWAN;. 6. Parâmetros de Transmissão;. 7. Algoritmos de Aprendizagem por Reforço;. I. C. de Lamare, Rodrigo. II. Longoni Madruga, Ewerton. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharía Elétrica. IV. Título.

CDD: 620.11

Acknowledgments

First of all, I wish to acknowledge the support of all my family, especially the dedication and the great love that I have always received from my parents. They give me all the confidence and comprehension I need to fight for my dreams. They are my strength, my guide, the engine of my life.

Thanks to my brother for his love and happiness, for making my day with your calls, and for taking care of our parents during this time that I have been away.

Then I wish to thank one of the most lovely, comprehensive, and intelligent men I know: my husband. Thanks for putting a smile on my face every day, for being my friend, my inspiration, for enjoying my dreams, for making it real. Thanks for the immense help with this thesis and for always being my safe place.

Thanks to my husband's family, especially to his parents, his little sister, and his grandfather Ubaldo. They have made me part of the family since day one, always offering me their love and unconditional support.

I want to make a special thanks to Dianelis, for taking care of my family as hers, for her comprehension and support.

I am so grateful to my advisor, Professor Rodrigo de Lamare, for trusting me and giving me the opportunity of doing this interesting work. He has provided me excellent guidance during my master's studies, contributing to my formation, and to accomplish the goals of this thesis.

Also, I want to thank to my co-advisor, Ewerton Madruga, who has guided my programming studies, giving me immense support with simulations and each time I needed his help.

I wish to express my deepest gratitude to my friend Julio and his future wife, Carina. Thanks for their support, advice, and for taking care of us when we arrive here. Also, thanks to the Cuban family I found here. In special to Kirenia, David, Lorena, Rocio, Emilio, Daylis, Randy, Adila, and Félix, and to those that arrive later, Madaine, Raul, Gretel, and Anabel. Thanks for being my family in Rio de Janeiro and for being there for everything.

I want to thank to my long-life friends Lisandra and Diego, and to my friends Liz, Claudia, Dalier y Lieter. They are always near despite the distance, giving me their support and love.

I would also like to thank to my colleagues and friends from CETUC, for all the moments that we share together, especially to Diana, Azucena, Jose, Juampi, and Alireza. Thanks to Marcelo Balisteri for his friendship, his jokes, for keeping me up to date with new technologies and for being the first to tell me about LoRaWAN.

I am truly thankful to Pontifical Catholic University of Rio de Janeiro, and especially to CETUC, for giving me the opportunity of pursuing my master's degree and for the excellent professors I had here. Thanks to the professors Raymundo and José Mauro for their dedication and excellent lessons.

I am truly thankful to Pontifical Catholic University of Rio de Janeiro, and especially to CETUC for giving me the opportunity of pursuing my master's degree in one of the most prestigious universities of Latin America. Thanks to the excellent professors I had here, especially to the professors Raymundo and José Mauro for their dedication, help, and excellent lessons.

Finally, I want to thank to the Brazilian Agency FAPERJ, which has supported this work under grant E-26/200.822/2019.

Abstract

Frómeta Fonseca, Dayrene; C. de Lamare, Rodrigo (Advisor); Longoni Madruga, Ewerton (Co-Advisor). **Machine Learning-Based MAC Protocols for LoRa IoT Networks**. Rio de Janeiro, 2020. 126p. Dissertação de mestrado – Departamento de Engenharía Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

With the massive growth of the Internet of Things (IoT), novel wireless communication technologies have emerged to address the long-range, low-cost, and low-power consumption requirements demanded by the IoT applications. In this context, the Low Power Wide Area Networks (LPWANs) have appeared, offering different solutions that meet the IoT applications' requirements mentioned before. Among the existing LPWAN solutions, LoRaWAN has stood out for receiving significant attention from both industry and academia in recent years. Although LoRaWAN offers a compelling combination of long-range and low-power consumption data transmissions, it still faces several challenges in terms of reliability and scalability. However, due to its open-source nature and the flexibility of the modulation scheme it is based on (Long Range (LoRa) modulation allows the adjustment of spreading factors and transmit power), LoRaWAN also offers important possibilities for improvements. This thesis takes advantage of the appropriateness of the Reinforcement Learning (RL) algorithms for solving decision-making tasks, and use them to dynamically adjust the transmission parameters of LoRaWAN end devices. The proposed system, called RL-LoRa, shows significant improvements in terms of reliability and scalability when compared with LoRaWAN. Specifically, it decreases the average Packet Error Ratio (PER) of LoRaWAN by 15%, which can further increase the network scalability.

Keywords

Low Power Wide Area Networks; Medium Access Control Protocols; LoRa Modulation; LoRaWAN; Transmission Parameters; Reinforcement Learning Algorithms;

Resumo

Frómeta Fonseca, Dayrene; C. de Lamare, Rodrigo; Longoni Madruga, Ewerton. **Protocolos MAC baseados em Aprendizado de Máquina para Redes de Internet das Coisas do Tipo Lora**. Rio de Janeiro, 2020. 126p. Dissertação de Mestrado – Departamento de Engenharía Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

Com o rápido crescimento da Internet das Coisas (IoT), surgiram novas tecnologias de comunicação sem fio para atender aos requisitos de longo alcance, baixo custo e baixo consumo de energia exigidos pelos aplicativos de IoT. Nesse contexto, surgiram as redes de longa distância de baixa potência (LPWANs), as quais oferecem diferentes soluções que atendem aos requisitos dos aplicativos de IoT mencionados anteriormente. Entre as soluções LPWAN existentes, o LoRaWAN tem-se destacado por receber atenção significativa da indústria e da academia nos últimos anos. Embora o LoRaWAN ofereça uma combinação atraente de transmissões de dados de longo alcance e baixo consumo de energia, ele ainda enfrenta vários desafios em termos de confiabilidade e escalabilidade. No entanto, devido a sua natureza de código aberto e à flexibilidade do esquema de modulação no qual ele se baseia (Long Range (LoRa) permite o ajuste de fatores de espalhamento e a potência de transmissão), o LoRaWAN também oferece importantes possibilidades de melhorias. Esta dissertação aproveita a adequação dos algoritmos de Aprendizagem por Reforço (RL) para resolver tarefas de tomada de decisão e os utiliza para ajustar dinamicamente os parâmetros de transmissão dos dispositivos finais LoRaWAN. O sistema proposto, chamado RL-LoRa, mostra melhorias significativas em termos de confiabilidade e escalabilidade quando comparado ao LoRaWAN. Especificamente, diminui a taxa de erro de pacote (PER) média do LoRaWAN em 15 %, o que pode aumentar ainda mais a escalabilidade da rede.

Palavras-chave

Redes de Longa Distância de Baixa Potência; Protocolos de Controle de Acesso ao Médio; Modulação LoRa; LoRaWAN; Parâmetros de Transmissão; Algoritmos de Aprendizagem por Reforço;

Table of contents

List of figures

List of tables

“We must have perseverance and above all confidence in ourselves. We must believe that we are gifted for something and that this thing must be attained”

Marie Curie

List of Abreviations

3GPP – Third Generation Partnership Project

ACK – Acknowledgment
ADR – Adaptive Data Rate
AWGN – Additive White Gaussian Noise

BER – Bit Error Rate
BPSK – Binary Phase Shift Keying

CRC – Cyclic Redundancy Check
CSS – Chirp Spread Spectrum

D2D – Device-to-Device
DBPSK – Differential Binary Phase Shift Keying
DP – Dynamic Programing
DSSS – Direct Sequence Spread Spectrum

FDMA – Frequency Division Multiple Access
FHSS – Frequency Hopping Spread Spectrum
FSK – Frequency Shift Keying

GSM – Global System for Mobile communications

IP – Internet Protocol
IoT – Internet of Things
ISM – Industrial, Scientific, and Medical

LoRa – Long Range
LPWAN – Low Power Wide Area Network
LTE – Long Term Evolution

MAB – Multi-Armed Bandit
MAC – Medium Access Control

MoT – MAC on Time
MDP – Markov Decision Process
ML – Machine Learning

NACK – No Acknowledgment
NB-IoT – Narrow Band IoT
NOMA – Non-Orthogonal Multiple Access
NS-3 –Network Simulator 3

OFDMA – Orthogonal Frequency Division Multiple Access
OSI – Open System Interconnection

PER – Packet Error Ratio
PDR – Packet Delivery Ratio

QPSK – Quadrature Phase Shift Keying
QoS – Quality of Service

RFTMA – Random Frequency and Time Multiple Access
RL – Reinforcement Learning
RL-LoRa – Reinforcement Learning LoRa
RSSI – Received Signal Strength Indicator

SIC – Successive Interference Cancellation
SINR – Signal-to-Interference-Noise Ratio
SF – Spreading Factor
SL – Supervised Learning
SNR – Signal-to-Noise Ratio SS – Spread Spectrum

TCP – Transmit Control Protocol
TD – Temporal-Difference

UCB – Upper Confidence Bound
USL – Unsupervised Learning

WLAN – Wireless Local Area Network
WSN – Wireless Sensor Networks

1
Introduction

Throughout the last few years, the IoT technologies have improved the way we live and work, becoming strong potential solutions to address the challenges that humanity is facing today, such as population growth, energy crisis, resource depletion, and environmental pollution. The full potential of the IoT networks can be exploited through IoT applications, which depending on their type, impose specific requirements in terms of data rates, power consumption, coverage area, and cost [6].

IoT applications for sectors such as transportation, healthcare, agriculture, and industry, require low data rates and long-range communications, as well as low-cost and low-power consumption end devices. Traditional wireless technologies, such as short-range wireless networks (e.g., ZigBee and Bluetooth), Wireless Local Area Networks (WLANs) (e.g., WiFi), and cellular networks (e.g., 2G, 3G, and 4G), are not able to satisfy all these requirements. The short-range wireless networks and WLANs are not adapted for scenarios that require long-range communications since their coverage is limited to a few hundred meters at best. On the other hand, solutions based on legacy cellular networks can provide larger coverage, but they have limitations in terms of energy efficiency, draining the batteries of end devices very quickly. Although 5G technology offers a modern network architecture, improving the energy efficiency for device-to-device (D2D) communication, it does not guarantee the requirement of low-cost end devices demanded by several IoT applications [6,7].

In this context, the LPWANs have appeared as a novel communication paradigm to complement and sometimes supersede traditional wireless technologies in addressing the IoT applications' requirements. LPWANs are intended to provide low-power (μW) and long-range (1–5 km in urban zones) communications to a massive number of nodes (thousands of end devices), at the expense of low data rate (thousands of bits per second) and high latency (seconds or minutes) [7]. Due to their features, LPWANs are highly suitable for IoT applications that run on battery-constrained end devices and need to transmit tiny amounts of data (hundreds of bytes) over a large geographical area.

1.1 LPWANs: typical setup and existing MAC layer solutions

Fig. 1.1 shows a typical LPWAN, which consists of nodes, gateways, and a network server. In this scenario, the IoT applications run at nodes, which can be sensors/actuators that send/receive data to/from the network server through the gateways. Nodes can be placed anywhere within gateways' coverage area and can be static or mobile.

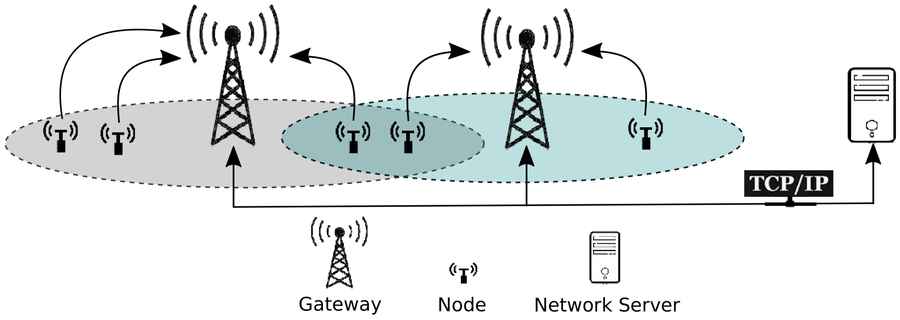

Figure 1.1: A typical Low Power Wide Area Network.

Gateways encapsulate the received messages from nodes in Transmit Control Protocol/Internet Protocol (TCP/IP) packets and transmit them to the network server. Multiple gateways can collaborate with each other, improving communication performance. That is, gateways will send the received messages to the network server even if these messages were already received by other gateways. The network server hosts all the intelligence in the network. It is responsible for orchestrating and monitoring different gateways, interpreting and collecting the data, removing duplicated messages, and forwarding the data to the corresponding destination (other application servers) if necessary.

Recently, many LPWAN technologies have arisen, each employing different techniques to achieve long-range, low power consumption, and low cost. Among the existing solutions, Sigfox [8], LoRa [9], and Narrow Band IoT (NB-IoT) [10] are today's leading LPWAN technologies, competing as underlying networking solutions for a variety of long-range IoT applications. These technologies involve many different aspects, not only economically (open-source versus closed source), but also in terms of reliability, scalability, coverage area, and power consumption [5].

As depicted in Table 1.1, different modulation techniques have been used at the physical layer of the existing LPWANs, including Chirp Spread Spectrum (CSS), Binary Phase Shift Keying (BPSK), and Quadrature Phase Shift Keying (QPSK). Similarly, different bandwidths (narrow or wideband) and frequency bands (licensed or unlicensed) have been employed.

Table 1.1: Physical layer features of LoRa, Sigfox, and NB-IoT [5].

Technology	**Modulation**	**Frequency**	**Bandwidth**
LoRa	CSS	Unlicensed ISM bands	125 kHz and 250 kHz
SigFox	BPSK	Unlicensed ISM bands	100 Hz
NB-IoT	QPSK	Licensed LTE frequency bands	200 kHz

It is known that the physical layer is essential to achieve long-range communications. However, even when the physical layer can provide the required distance, the Medium Access Control (MAC) layer plays a crucial role. It has a profound impact on network performance since it can increase network reliability and scalability by avoiding packet collisions through implementing efficient channel access control mechanisms. Besides, the MAC layer is in charge of determining when nodes should transmit, receive, listen, or simply remain idle. It thus influences the energy consumption of the end devices, which is critical in the context of IoT networks.

In the past few years, several MAC layer solutions for the LPWANs have been proposed, each trying to address the IoT applications' requirements by employing different approaches. Many proposals rely on pure ALOHA-based channel access, such as LoRaWAN [4] and Random Frequency and Time Multiple Access (RFTMA) [11], the MAC layer protocols used by LoRa and SigFox based networks, respectively. ALOHA-based approaches have the advantage of allowing low power consumption for nodes that have a very low demand on throughput. However, they can not avoid packet collisions, which considerably affects the network performance in terms of reliability and scalability. Other proposals are based on synchronizing the communication between gateways and end devices. Such is the case of Symphony Link [12] and MAC on Time (MoT) [13], two synchronous MAC protocols built to operate on top of the LoRa physical layer. Finally, some approaches employing Successive Interference Cancellation (SIC) techniques have also been reported. An example is ALOHA-NOMA [14], a new scalable and energy-efficient MAC protocol that combines pure ALOHA with power domain non-orthogonal multiple access (NOMA) to increase the throughput in IoT networks.

Among the proposed MAC layer solutions for LPWANs, LoRaWAN has stood out for receiving significant attention from both industry and academia in recent years [15, 16]. Although to date, LoRaWAN is not the best MAC protocol for LPWANs in its simplified model [17], it shows interesting opportunities for improvement due to its *open-source* nature. Moreover, the wide availability of cheap development boards and open-source code [18], has made this protocol an excellent candidate for studying long-range communications.

1.2 Limitations of LoRaWAN

Since the LoRaWAN protocol was designed to work on top of the LoRa physical layer, it takes advantage of this Spread Spectrum (SS) modulation technique. LoRa modulation can enhance the network capacity by using orthogonal Spreading Factors (SF), which enables the simultaneous transmission of multiple signals with different SFs. Other benefits of LoRa are its long-range capability, and its robustness to interference, multipath, and fading, making it ideal for use in urban and suburban environments [19].

Despite the advantages derived from the LoRa physical layer, LoRaWAN still faces several challenges in terms of reliability and scalability. The reason is that it has limited choices available for power and spreading factor control: five different values for power (2, 5, 8, 11 and 14 dBm) and six unique SFs, going from SF7 (the highest data rate) to SF12 (the lowest data rate) [4]. This limited set of SFs makes LoRaWAN networks very prone to packet losses due to packet collisions (only six nodes using different SFs can transmit simultaneously on the same channel without causing collisions), which greatly degrades the network performance in terms of reliability. Furthermore, multiple packets with different SFs can also collide, resulting in a bad Signal-to-Interference-Noise Ratio (SINR) for all packets involved. A packet suffering this kind of collision can also be dropped, which occurs if it collides with a more stronger packet [20]. Thus, in LoRaWAN networks, it is essential to combine power and SF control to allow more reliable communications.

Although LoRaWAN implements the Adaptive Data Rate (ADR) mechanism to adapt the nodes' transmit power and SF locally, it results in an unfair network with a high PER for nodes far from the central gateway [20]. This is because LoRaWAN networks are sensitive to the *capture effect*[1], which lets high powered packets survive collisions, while low powered packets get lost. As a result, according to the ADR mechanism, nodes that are further from the gateway will see significantly more collisions, increasing their SF to increase the link budget [4]. However, it only will increase the number of collisions because higher SFs consume more time on-air and produce more interference to the network [20].

[1] The capture effect is a phenomenon associated with the reception in frequency modulation, in which only the stronger of two signals at, or near, the same frequency or channel will be demodulated.

1.3 Evaluating and improving LoRaWAN's performance: challenges and related work

Recently, there has been an increasing interest from the research community not only to exploit the range and low-power of LoRaWAN networks but also to assess and improve LoRaWAN's reliability and scalability. In that regard, a large number of studies have been carried out to evaluate the performance of this protocol, both conceptually and through simulations. A comprehensive analysis of the capabilities and limitations of LoRaWAN can be found in [21]. The works in [22–24] investigate the scalability of LoRaWAN, showing that the Packet Delivery Ratio (PDR) decreases exponentially with the number of nodes in the network. In [25], it is also conducted a scalability analysis of LoRaWAN, but focusing on the effect of confirmed versus unconfirmed traffic. The authors in [17] evaluate LoRaWAN's performance in scenarios with both single and multiple gateways, concluding that acknowledgments are not scalable and that multiple gateways improve the network reliability considerably. Finally, the LoRaWAN protocol has also been evaluated under mobile scenarios, such as in [26–28]. All the mentioned works demonstrate the worsening of the network performance in terms of packet losses when increasing the number of nodes, exposing the reliability and scalability limitations of LoRaWAN.

Considerable research work has also been done to improve LoRaWAN's scalability and reliability [29]. Some approaches are based on algorithms for appropriately selecting SFs and transmit powers, which have appeared in two variants: selection algorithms to be executed (1) at the nodes' side [30–32], or (2) at the network server or gateways [20,33,34]. There are also some proposals to improve the ADR mechanism of LoRaWAN, such as the works presented in [35] and [36].

These approaches to efficiently distribute SFs and transmit powers and to enhance the LoRaWAN's ADR mechanism can improve network scalability. However, they can not solve the main network scalability bottleneck: the use of an ALOHA based MAC protocol. To this end, other solutions to coordinate transmissions between nodes and gateways have been studied [37–41]. Such solutions rely on synchronization approaches and low-overhead traffic scheduling mechanisms, to accomplish the duty cycle restrictions of LoRaWAN.

More recently, researchers have explored RL techniques to improve the performance of LoRaWAN. The works in [42] and [43] proposed RL approaches hosted by LoRaWAN nodes to dynamically adjust their transmission parameters (SFs and transmit power). Both of them have improved LoRaWAN's performance by minimizing energy consumption and collision probability. They are two of the earliest works providing adaptive transmission parameters allocation in LoRaWAN, and their simulations show great results compared to the state-of-the-art approaches, which is expected since learning-based approaches allow nodes to adapt their transmission parameters according to the network dynamics: the number of nodes can increase/decrease, interfering signals can appear/disappear, and the end devices' traffic patterns can change. However, in practical applications, the results presented in [42] and [43] will be considerably worse. This is because their proposal relies on Acknowledgment (ACK) or No Acknowledgment (NACK) feedback, which is very challenging in practice because the interference generated by ACK signals will lead to high packet losses in the network. This effect is not reflected in their simulation results because their proposal was evaluated using a perfect downlink feedback channel, overlooking the effect of the interference generated by the ACK signals.

According to the above, using learning approaches for dynamically adjusting the transmission parameters at the end devices looks like a prominent approach to optimize the performance of LoRaWAN. In this sense, RL algorithms look like an excellent tool for designing such algorithms. This is because adjusting the transmission parameters at the end devices can be seen as a decision-making task where nodes have to learn the best set of parameters for transmitting their data, and RL algorithms are ideal for solving this kind of task [44]. However, it is clear that designing RL algorithms for locally controlling the transmission parameters of LoRaWAN end devices is very challenging. The main reason is the limited feedback that LoRaWAN offers for rating the decisions made by the RL algorithms running at the nodes' side. In this way, implementing RL algorithms at the end devices' side requires efficient methods for carrying the reward information from gateways to the nodes. That is, without affecting the network performance by adding extra interference or increasing the packet losses.

To conclude, one can say that although LoRaWAN faces several challenges in terms of scalability and reliability, it also shows interesting opportunities for improvement. However, to leverage these opportunities, a better understanding of the LoRa physical layer and the LoRaWAN MAC layer will be necessary. Likewise, it is worth highlighting that RL approaches look like prominent solutions to dynamically adapt the transmission parameters of the end devices in LoRaWAN networks [42]. Nevertheless, these learning methods expect rewards for their selected actions, which is hard to achieve in LoRaWAN due to the limited amount of downlink messages it offers.

1.4 Objectives of the dissertation

Motivated by the above state of the art, this thesis aims to improve LoRaWAN's scalability and reliability by applying RL techniques, defining the following research objectives to this end:

1. *To analyze the main features of LoRa modulation and LoRaWAN MAC layer.* It will be essential to understand the performance of LoRaWAN networks and to identify possible gaps for improvement.

2. *To study the fundamental concepts and algorithms of Reinforcement Learning.* A solid understanding of the RL basics, as well as its algorithms, will be necessary to develop methods that effectively improve the performance of LoRaWAN.

3. *To develop RL based algorithms that improve the performance of LoRaWAN networks by dynamically adjusting the transmission parameters of the end devices.* The transmission parameters adaptation in LoRaWAN networks can be modeled as an RL problem, and different RL algorithms can be used to solve it. This thesis aims to identify which of these algorithms brings better results when adjusting the transmission parameters of LoRaWAN end devices.

4. *To propose efficient mechanisms for carrying from gateways to the end devices the reward information required by the RL algorithms that are running at nodes.* One of the challenges associated to applying RL techniques for locally adjusting the transmission parameters of end devices, is the limited feedback LoRaWAN offers to rate the decisions made by the RL algorithms that are running at nodes. In that sense, to realize the full potential of the proposed RL algorithms to adapt the nodes' transmission parameters, it will be necessary to develop efficient mechanisms for carrying the reward information from gateways to end devices.

5. *To evaluate through simulations the proposed RL algorithms, and to analyze the performance improvements they bring to LoRaWAN networks in terms of scalability and reliability.* Finally, this thesis aims to carry out a performance comparison between the proposed RL algorithms and state-of-the-art proposals for improving the performance of LoRaWAN.

1.5 Major research contributions

The major contributions of this thesis are listed below:

1. *Extending the knowledge about the physical and MAC layer of LoRaWAN networks: the LoRa modulation and the LoRaWAN MAC protocol, respectively.*
   This thesis broadly describes the network architecture and protocol stack of LoRaWAN networks, paying special attention to the physical and MAC layers. It introduces a detailed description of the functionalities of these layers, explaining important mechanisms with a strong influence on the network performance, such as the ARD mechanism to control the transmission parameters at the end devices, and the rules to decide when a packet is lost as a consequence of a collision.

2. *Exposing the limitations of the LoRaWAN protocol.*
   After studying the physical and MAC layers of LoRaWAN, the next contribution of this work is an evaluation of the LoRaWAN MAC protocol in terms of important metrics in the context of the IoT applications, such as the average PER, network throughput, average network delay, and energy consumption, and considering scenarios with single and multiples gateways. This performance evaluation exposes the limitations of LoRaWAN and identifies possible optimizations for improving communications in LoRaWAN networks. The main results of this performance evaluation have been published in [28].

3. *Extending the LoRaWAN MAC layer and improving its performance*
   To enable the use of RL techniques for dynamically controlling the transmission parameters of LoRaWAN end devices, this thesis has proposed a new MAC protocol based on the legacy LoRaWAN. The new scheme, called RL-LoRa, uses RL algorithms for locally adjusting the SFs and transmit power at the end devices and beacons for carrying the required feedback from gateways to the RL algorithms that are running at nodes.

4. *Implementing the proposed RL-LoRa MAC protocol in NS-3*
   The last contribution of this thesis is an NS-3 module that implements the proposed RL-LoRa protocol. It provides base C++ classes that facilitate the implementation and integration of new RL algorithms to locally control the transmission parameters of LoRaWAN end devices.

1.6 Organization of the dissertation

The remainder of this thesis is organized as follows:

- *Chapter 2:* Low Power Wide Area Networks and Reinforcement Learning.

  This chapter describes the physical and MAC layers of three of the most well-known LPWAN solutions: LoRaWAN, SigFox, and NB-IoT, highlighting LoRaWAN as the solution that offers more opportunities for improvements. Then, it presents the fundamental concepts and algorithms of RL, identifying the UCB and Q-Learning algorithms as the most widely applied to solve problems related to communication networks.

- *Chapter 3:* Analyzing and Evaluating the Performance of LoRaWAN.

  This chapter includes a detailed analysis of the physical and MAC layer of LoRaWAN networks, describing important mechanisms that have placed at these layers as the ADR mechanism for controlling the transmission parameters at the end devices. It also evaluates LoRaWAN's performance under different scenarios and in terms of various metrics, identifying the limitations of this protocol when operating in a shared medium.

- *Chapter 4:* Improving the LoRaWAN MAC layer through Reinforcement Learning techniques.

  In this chapter, an extension of the LoRaWAN MAC layer is proposed, which improves network performance by using RL techniques to control the transmission parameters at the end devices. First, a detailed description of the proposed system is provided. Then, it is evaluated under different scenarios and compared with LoRaWAN and state-of-the-art proposals that try to improve the performance of LoRaWAN.

- *Chapter 5:* Conclusions and future work

2
Low Power Wide Area Networks and Reinforcement Learning

2.1
Low Power Wide Area Networks

In Section 1.1, the elements and typical setup of an LPWAN were introduced. This section first presents a formal definition for LPWANs and then describes the most well-known LPWAN solutions proposed by industry.

2.1.1
LPWAN: a definition

In [7], authors define LPWANs as “ wireless technologies that can offer low-power connectivity to a massive number of devices distributed over large geographical areas at an unprecedented low cost”. Below, it is discussed how LPWANs can meet the goals involved in this definition.

- **Long Range:** In LPWANs, nodes can reach gateways at a distance ranging from a few to tens of kilometers depending on their deployment environment (rural, urban, etc.). Special modulation schemes (narrowband and SS), and operation on the Sub-GHz band have been adopted by different LPWAN technologies to accomplish this goal [7]. This is because narrowband modulations provide a high link budget by encoding the signal in low bandwidth, while SS techniques are more robust and resilient to interference signals. On the other hand, operating on the Sub-GHz band allows better propagation characteristics (there are less attenuation and multipath fading at lower frequencies), and lower congestion compared to the overly popular 2.4 GHz Industrial, Scientific, and Medical (ISM) band.
- **Ultra Low Power Operation:** It is a key requirement to take advantage of the excellent business opportunity provided by battery-powered IoT devices. To this end, different duty cycling mechanisms have been adopted by the existing LPWAN solutions, which can vary depending on application requirements, type of power source, and traffic pattern, among other parameters. Other popular approaches applied by almost all the LPWANs to achieve low power consumption, are the use of star

topology and simple random access schemes. This is because, in a star topology, devices do not need to waste energy in listening to other devices that want to deliver their traffic through them. Likewise, simple MAC schemes do not introduce excessive signaling overhead, which brings huge energy-saving advantages [7].

- **Low Cost:** LPWAN technologies have adopted different ways to reduce the cost for both the end-users and network operators. Low-cost end devices are possible thanks to several techniques, some of which were already discussed in this section. The use of star (instead of mesh) topology, and simple MAC protocols, enables manufacturers to design simple and therefore low-cost end devices. On the other hand, to reduce the cost for network operators, many LPWANs consider deployments in license-free frequency bands, such as the ISM band [7].
- **Scalability:** Several techniques have been considered to allow LPWANs working well when increasing the number and density of connected devices. Two examples are dense deployments of gateways and mechanisms for adaptively adjusting the modulation schemes. Also, in cases when some downlink communication is possible, gateways and network servers can play a vital role in selecting optimal transmission parameters to increase reliability, which further improves network scalability. [7].

To conclude, it is important to highlight that adapting the modulation schemes, selecting better channels to reach distances reliably, or adaptively adjusting the transmission parameters, require efficient monitoring of the link quality and synchronization between gateways and end devices. Generally, it implies more complex end devices and extra communication overhead over the uplink and the downlink, resulting in higher energy consumption and more expensive end devices. Thus, in LPWANs, there is a clear trade-off between network scalability, energy consumption, and the simplicity of low-cost end devices.

2.1.2 LPWAN solutions

A complete description of different LPWAN solutions proposed by industry in the past few years can be found in [6,7]. Although many other solutions could be presented here, this section only introduces the main features of LoRaWAN, SigFox, and NB-IoT. This is because they are three of the most well-known LPWAN solutions, constituting clear application examples of the techniques discussed in Section 2.1.1 to achieve the LPWANs' main goals.

2.1.2.1 LoRaWAN

LoRaWAN is an open-source MAC protocol standardized by the LoRa Alliance [45] and designed to run on top of the LoRa physical layer, as depicted in Fig. 2.1.

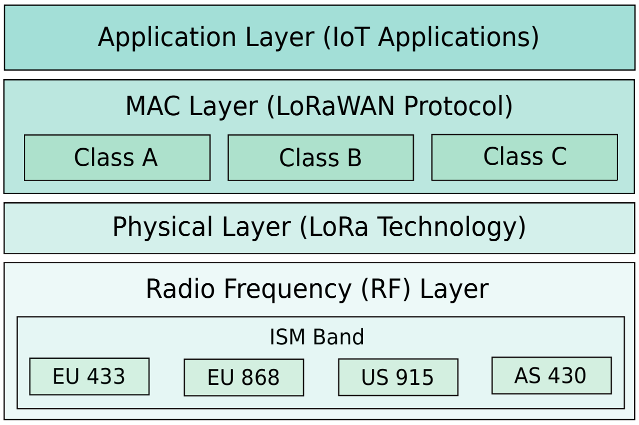
Application Layer (IoT Applications)
MAC Layer (LoRaWAN Protocol)
Class A
Class B
Class C
Physical Layer (LoRa Technology)
Radio Frequency (RF) Layer
ISM Band
EU 433
EU 868
US 915
AS 430

Figure 2.1: The network stack of LoRaWAN [1].

LoRa modulation is based on the CSS scheme, which uses wideband linear frequency modulated chirp[1] pulses to encode information. As other SS methods, such as Direct Sequence Spread Spectrum (DSSS) or Frequency Hopping Spread Spectrum (FHSS), LoRa uses the entire allocated bandwidth to broadcast the signal, making it robust to channel noise, multi-path fading and Doppler effect [19]. LoRa's symbol modulation is based on the SF value, which ranges from 7 to 12 if LoRaWAN is used as the MAC layer [4]. The higher SF corresponds to the longer symbol time and the lower data rate but also the most robust symbol. When using the LoRaWAN MAC protocol, the LoRa data rate is between 300 bps and 50 kbps depending on the SF and channel bandwidth, which can be 125, 250, or 500 kHz. Finally, as can be observed in Fig. 2.1, LoRa modulation handles the communication between gateways and end devices in different sub-GHz frequency bands, depending on local frequency regulations.

The LoRaWAN MAC protocol uses an ALOHA based uplink, and its downlink consists of two slots that are opened after each uplink transmission. Depending on the application scenario, LoRaWAN allows nodes to be configured into three types: Class A, B, or C, as shown in Fig. 2.1. Each of these classes constitutes a trade-off between energy consumption and network downlink communication delay, being Class A, the one that consumes the least amount of energy [4]. On the other hand, in LoRaWAN, the maximum payload

[1] A chirp pulse is a sinusoidal signal whose frequency increases or decreases over time.

size depends on the SF used, going from 51 bytes (for SF12) up to 242 bytes (for SF7) according to the LoRaWAN specifications [4].

It is important to clarify that this thesis addresses the LoRaWAN[2] operation according to the European regulations, which states that LoRaWAN end devices should be capable of operating in the 863 to 870 MHz frequency band, and should feature a channel data structure[3] to store the parameters of at least 16 channels. It also establishes three default channels that must be implemented in every LoRaWAN node, corresponding to 868.1, 868.3, and 868.5 MHz, each one allowing data rates from DR0 (SF12) to DR5 (SF7) [4]. In addition to the LoRa based channels, LoRaWAN also allows a single uplink/downlink channel based on the Frequency Shift Keying (FSK) modulation, which enables data transmissions at 50 kbps [1, 4].

Finally, since this thesis focuses on improving the performance of LoRaWAN based networks, a more detailed description of LoRa modulation and the LoRaWAN MAC protocol is going to be presented in Appendix A and Chapter 3, respectively.

2.1.2.2 SigFox

SigFox is a proprietary LPWAN solution owned by the SigFox company [8]. As LoRaWAN, SigFox operates on the unlicensed ISM band and transmit data in an ALOHA fashion. However, due to its closed infrastructure, SigFox does not allow too many opportunities for performance improvements as LoRaWAN does.

SigFox is a bidirectional technology with a significant link asymmetry [5]. Its uplink uses an ultra-narrowband (100 Hz) modulation called Differential Binary Phase Shift Keying (DBPSK), which enables very low noise levels and high receiver sensitivity, leading to a range between 10 and 50 kilometers for urban and rural environments, respectively [6, 7]. On the other hand, the uplink of SigFox only transfers data with a fixed data rate of 100 bps and limits the number of transmitted messages per day and the payload size to 140 messages and 12 bytes, respectively [5].

The SigFox's downlink is relatively faster (600 bps) than the uplink. It dedicates a separate channel for downlink communication (data from the gateways to the end devices) and establishes that a downlink communication

[2] The remainder of this document uses the term LoRaWAN to refer to the combination of the LoRaWAN MAC protocol with the LoRa physical layer. When talking about the physical layer, it will be used LoRa.

[3] A channel data structure corresponds to a frequency and a set of data rates usable on that frequency [4].

can only occur following an uplink communication [7]. As the uplink, the downlink of SigFox also limits the number of transmitted messages per day and the payload size to 4 messages and 8 bytes, respectively [5]. This limited number of downlink messages means that acknowledging every uplink message in SigFox based networks is impossible.

The SigFox's uplink MAC protocol is called RFTMA. Using it, the transmitter selects at random a central frequency from the allowed bandwidth and sends its messages without sensing to the receiver. By randomizing the central frequency combined with the ultra-narrowband signal, SigFox assumes that the collision probability remains very low [46].

Finally, it is important to highlight that due to the small message sizes and the limited number of allowed packet transmissions per day, SigFox based networks consume less energy than other LPWAN solutions, consequently prolonging the battery life of its end devices. Moreover, since SigFox uses a separate channel for downlink, uplink and downlink messages will never interfere with each other.

2.1.2.3 NB-IoT

NB-IoT is a narrow-band technology specified in Release 13 of The Third Generation Partnership Project (3GPP) [10]. It is based on the Long Term Evolution (LTE) protocol, adapting the latter's functionalities to the requirements of IoT applications. NB-IoT can coexist with Global System for Mobile communications (GSM) and LTE under licensed frequency bands (e.g., 700, 800, and 900 MHz), and occupies a frequency bandwidth of 200 kHz [5].

NB-IoT allows connectivity of up to 50 K end devices per cell with the potential for scaling up the capacity by adding more NB-IoT carriers. It employs QPSK modulation and defines a maximum payload size for each message of 1600 bytes. Like other LPWAN solutions, NB-IoT also limits the allowed data rate, establishing 20 kbps for uplink and 200 kbps for downlink. On the other hand, NB-IoT uses different MAC protocols for uplink and downlink, being single-carrier Frequency Division Multiple Access (FDMA) for the uplink and Orthogonal FDMA (OFDMA) for the downlink [10].

To conclude, one can say that NB-IoT meets the main goals of the LPWANs by supporting massive connections, enabling prolonged battery life (NB-IoT end devices can achieve ten years of battery lifetime when transmitting 200 bytes per day on average), and providing long-range (1 km and 10 km for urban and rural deployments, respectively). Although its first release (R13) presented some limitations, including a low localization accuracy, low data rates, and no support for mobility, more features have been added to NB-IoT in 3GPP LTE Release 14, providing better performance while maintaining the R13 current merits [6].

Table 2.1 summarizes the technical aspects of LoRaWAN, Sigfox, and NB-IoT that have been discussed in this section.

Table 2.1: Comparison among the LPWAN solutions: LoRaWAN, SigFox, and NB-IoT.

	LoRaWAN	**SigFox**	**NB-IoT**
Modulation	CSS and FSK	DBPSK	QPSK
Frequency	Unlicensed ISM bands (868 MHz in Europe, 915 MHz in North America, and 433 MHz in Asia)	Unlicensed ISM bands (868 MHz in Europe, 915 MHz in North America, and 433 MHz in Asia)	Licensed LTE frequency bands (700 MHz, 800 MHz, and 900 MHz)
Bandwidth	125 kHz, 250 kHz, and 500 kHz	100 Hz	200 kHz
Maximum Data Rate	50 kbps	100 bps (uplink) 600 bps (downlink)	20 kbps (uplink) 200 kbps (downlink)
Adaptive Data Rate	Yes	No	No
Maximum Payload Length	242 bytes	12 bytes (uplink) 8 bytes (downlink)	1600 bytes
MAC Protocol	ALOHA	RFTMA	FDMA/OFDMA
Maximum messages/day	Unlimited	140 (uplink) 4 (downlink)	Unlimited
Topology	Star	Star	Star
Range	5 km (urban) 20 km (rural)	10 km (urban) 50 km (rural)	1 km (urban) 10 km (rural)
Standardization	LoRa-Alliance	SigFox Company	3GPP

2.2 Machine Learning for communication networks

In the past years, Machine Learning (ML) has gained popularity due to the variety of complex problems that can be solved by it. Some application examples are facial recognition, social media services, and product recommendations, among others. Specifically, ML focuses on the development of computer programs (algorithms and models) that can learn to make better decisions in the future based on the recognition of patterns in the provided data (datasets). In this way, computers can learn to adjust their actions without any human intervention or assistance.

In ML, there are three basic learning paradigms: Supervised Learning (SL), Unsupervised Learning (USL), and RL, as well as four broad categories of tasks that can be solved, namely, classification, regression, clustering, and rule extraction [47]. SL uses labeled training datasets to conduct classification and regression tasks, whereas USL uses unlabeled training datasets to create models that can classify the sample sets into different groups, being more suited for clustering problems [44]. While in SL and USL, the learning is based on samples from training datasets, the RL algorithms (agents) learn by interacting with the external world (environment) and exploiting the knowledge. The agents take actions that are rewarded or penalized and use this feedback from the environment to learn the best sequence of actions that optimize a cumulative reward. Due to the fact that RL algorithms may sacrifice immediate gains for long-term rewards, they are best suited for decision-making, planning, and scheduling tasks [47].

All the ML paradigms have been applied to fundamental problems in networking, including traffic classification, prediction and routing, resource management, congestion control, and network security [47]. However, in the past few years, RL has drawn the most attention due to the ease of use of its algorithms and because it can help in decision-making tasks, facilitating network scheduling, parameter adaptation, and resource allocation maximization under dynamic environments.

Researchers have explored various RL approaches to address common issues in wireless communication networks. Currently, there are many proposals using Multi-Armed Bandit (MAB) algorithms and Q-Learning to enhance the performance of MAC protocols for Wireless Sensor Networks (WSN) [48]. Some examples are RL-MAC [49], ALOHA-QIR [50], ALOHA-Q [51], and QL-MAC [52], which use Q-Learning based algorithms to manage the medium access adaptively. These protocols can find an efficient radio sleep-wakeup schedule and synchronize the nodes' transmission in a frame-based structure, considering parameters such as the traffic load and the channel bandwidth. Other proposals are focused on reducing energy consumption and latency in WSNs. An example is SSA [53], a distributed scheduling approach based on Q-Learning that enables nodes to learn the transmission and sleep parameters through interacting with the WSN.

Recently, some proposals applying RL approaches to LPWANs have also been reported. In [54], authors show that RL algorithms can be used for frequency selection and latency reduction in LPWANs. The work in [42], presents a learning solution to adjust the communication parameters of IoT devices for maximizing energy efficiency and reliability in data transmissions.

In [55] are introduced different Q-learning methods to optimize the number of served IoT devices in NB-IoT networks, demonstrating that the proposed RL approaches significantly outperform conventional methods in terms of throughput. Finally, authors in [43] suggest that LoRaWAN nodes use light-weight learning methods, namely, MAB algorithms, to select the transmission parameters (SF and transmit power). Their simulation results show that the proposed learning approaches can manage the trade-off between energy consumption and packet loss much better than the ADR mechanism of LoRaWAN.

Since this thesis focuses on applying RL approaches to improve the performance of LoRaWAN, the remainder of this chapter introduces the fundamental concepts, elements, and algorithms of RL. Although many RL algorithms could be presented here [56], only Upper Confidence Bound (UCB) and Q-Learning are going to be described. The reason is that UCB and Q-Learning are the most widely used RL algorithms in the context of LPWANs, and also because the approaches proposed in this dissertation are based on them.

2.3 Reinforcement Learning

RL is an area of ML concerned with learning how to behave in order to maximize a numerical reward that expresses a long-term goal. It is goal-oriented learning, where learning occurs via interacting with an environment. The learner is not taught what actions to take; instead, it must discover which actions yield the highest reward by trying them [2, 56].

2.3.1 Elements of Reinforcement Learning

The main components of an RL system are shown in Fig. 2.2. As can be observed, it consists of an *agent*, an *environment*, a *policy*, a *reward signal*, a *value function*, and, optionally, a *model* of the environment [2].

Agents are basically the learners in RL. They are software programs that make intelligent decisions in order to achieve their goals. Agents take action by interacting with the environment and receiving rewards based on their actions. In general, actions are the decisions the agent wants to learn how to make, and everything outside the agent is called the *environment*.

After each taken action, the environment sends to the RL agent a single number called the *reward*. This reward signal defines the goal of an RL problem, determining which are the good and bad events for the agent. The main

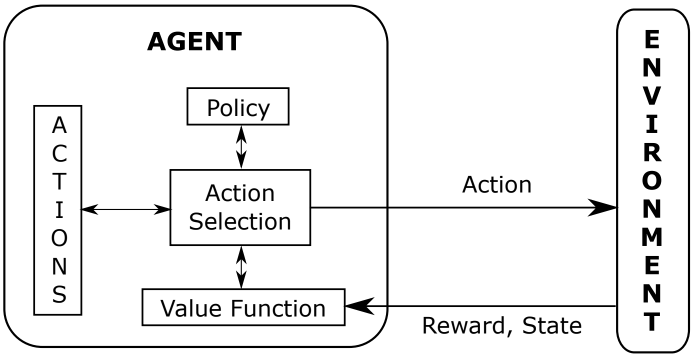

Figure 2.2: Elements of a Reinforcement Learning System.

objective of agents is to maximize the total reward they receive over the long run.

The *policy* is the core of an RL agent in the sense that it is enough to determine the agent's behavior at a given time. It establishes how the agent selects actions and can be seen as a mapping from perceived states of the environment to actions to be taken when in those states [2]. The policy may be a simple function or lookup table, or, in some cases, it can be much more complex such as a search process. In general, policies may be stochastic, specifying the probabilities of selecting each possible action [57].

A *value function* is a function of states (or of state-action pairs) that denotes how good it is for an agent to be in a particular state (or how good it is to perform a given action in a particular state). The value of a state corresponds to the total reward an agent expects to receive in the future, starting from that state. In this sense, without rewards, there are no values, and the objective of estimating values is to achieve more rewards. As depicted in Fig. 2.2, the action choices are made based on value judgments. The agent looks for actions that bring states of the highest value, instead of the highest reward, because these actions will obtain the greatest amount of reward over the long run. Estimating values is a complex process because they have to be estimated and re-estimated from the sequence of observations that an agent makes over its entire lifetime. In fact, a method for efficiently estimating values is the most important element of almost all the RL algorithms [2].

The final element of some RL systems is a *model* of the environment. It is the agent's representation of the environment, allowing to make inferences about how the environment will behave. Depending on the use or not of the environment's model, there are two types of RL methods: *model-based* methods and *model-free* methods, respectively [2].

2.3.2 Multi-Armed Bandit problem

MAB is a simplified setting of the RL problem that does not involve learning to act in more than one situation [2]. It is one of the classical problems in RL and can be described as follows.

Consider an agent that repeatedly has to choose among k different options or actions. After each choice, the agent receives a numerical reward selected from a stationary[4] probability distribution that depends on the chosen action. The agent's objective is to maximize the total expected reward over some period of time, for example, over 1000 action choices, or *time steps* [2].

Denoting the action selected at time step t as A_t, and the corresponding reward as R_t, the value of an arbitrary action a, denoted $q_*(a)$, can be defined as the expected reward given that a is selected, this is [2]:

$$q_*(a) = \mathbb{E}[R_t|A_t = a]. \tag{2-1}$$

If the agent knows the value of each action, it would be easy to solve the k-armed bandit problem: the agent would always select the highest value action. However, in general, agents do not know the action values. Instead, they should maintain estimates of these values so that at any time step, there is at least one action whose estimated value is highest. These actions are known as *greedy* actions, and when an agent selects one of them, it is said that the agent is *exploiting* its knowledge about the values of the actions. In contrast, if it chooses one of the *nongreedy* actions, it is said that the agent is *exploring* [57].

The trade-off between exploration and exploitation is one of the challenges that arise in RL. The agent must prefer *greedy* actions to obtain a lot of reward, but to discover such actions, it has to try actions that have not been selected before. Exploitation allows the agent to maximize the expected reward on a particular step, but exploration may produce the greater total reward in the long run.

There are many methods for balancing exploration and exploitation in the context of k-armed bandit problems, as well as different approaches for estimating the action values and for using these estimates to make action selection decisions [56]. Below are some of the most popular methods used for these purposes.

[4] It means that the reward probabilities do not change over time. According to this, the RL problems can be classified as *stationary* or *nonstationary* [2].

2.3.2.1 Action values estimation

A simple way to estimate the action values is by averaging the rewards each action has received so far. It is known in the literature as the *sample-average* method and defines the following expression for calculating the action values:

$$Q_t(a) = \frac{\sum_{i=1}^{t-1} R_i \times \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}}, \tag{2-2}$$

where $Q_t(a)$ denotes the estimated value of an action a at time step t, and $\mathbb{1}_{predicate}$ is a random variable whose value is '1' if *predicate* is true, or '0' if it is false [2].

From Eq. (2-2), it is possible to deduce an incremental implementation for the *sample-average* method, which will allow calculating the action value estimates in a computationally efficient manner. This incremental implementation is given by the following equation [2]:

$$Q_{n+1}(a) = Q_n(a) + \frac{1}{n}[R_n(a) - Q_n(a)], \tag{2-3}$$

where, $Q_n(a)$ denotes the estimated value for an action a after it has been selected $n-1$ times, and $R_n(a)$ is the reward received after the nth selection of this action.

The update rule in Eq. (2-3), can be presented in its general form as follows [2]:

$$NewEstimate \leftarrow OldEstimate + StepSize[Target - OldEstimate], \tag{2-4}$$

where, the expression [*Target* - *OldEstimate*] can be seen as an *error* in the estimate of the action value, which will be reduced as the action value estimate approaches the *Target*. The target indicates the desirable direction in which to move, and, in the case of the *sample-average* method, corresponds to the nth reward. The *StepSize* parameter changes from time step to time step, and, as can be observed in Eq. (2-3), for the *sample-average* method, it decreases with the the number of times an action has been selected (n).

Finally, it is important to highlight that the desired methods for estimating action values are those that let $Q_t(a)$ to be close to $q_*(a)$. In that sense, when estimating $Q_t(a)$ using the *sample-average* method, by the law of large numbers, $Q_t(a)$ will converge to $q_*(a)$ as the number of times that a has been selected goes to infinity. Moreover, it should be noticed that the *sample-average* method is just one way to estimate action values, and not necessarily the best one [2].

2.3.2.2 Action selection

Defining the set of available actions as $\mathcal{A} = \{a_0, a_1, ..., a_{I-1}\}$, the most straightforward action selection rule is to choose from $\mathcal{A}$ the action a_i with the highest estimated value. It is known as the *greedy* action selection method, which denotes the action selected at time step t as:

$$A_t = \underset{a_i}{\operatorname{argmax}}\, Q_t(a_i), \tag{2-5}$$

where $\operatorname{argmax}_{a_i}$ denotes the action a_i for which $Q_t(a_i)$ is maximized. If there is more than one greedy action, the selection among them is made at random. As can be noticed, the greedy action selection method never explores among the available actions; it always exploits the current knowledge without considering that other actions might be better in the long run [2].

An alternative to the previous approach is the ϵ-*greedy* selection method. It states that agents select the highest value actions with probability $1 - \epsilon$ and random actions with probability ϵ [57]. Thus, an epsilon value of zero will result in constantly exploiting, while a value of one will result in always exploring. The advantage of ϵ-*greedy* over *greedy* methods is that, in the limit, as the number of steps increases, every action will be selected an infinite number of times, ensuring that $Q_t(a_i)$ converge to $q_*(a_i)$ for all $i = 0, 1, ..., I - 1$ [2].

Another approach to address the exploration-exploitation problem in RL is known as the *decreasing* ϵ-*greedy* method. Basically, ϵ is initialized with a high value to allow more exploration at the beginning, and it is reduced at each time step for more exploitation. A simple expression for determining the value of ϵ in this case is given by:

$$\epsilon = \frac{c}{c + |t|}, \tag{2-6}$$

where c is a constant that controls the rate at which ϵ decreases, and t represents the current step of the RL algorithm. It can be noticed that ϵ will decrease when the number of steps increases, converging to zero when t is high enough.

2.3.2.3 The Upper Confidence Bound algorithm

When using the *ϵ-greedy* and *decreasing ϵ-greedy* methods, the agent explores random actions with a probability, with no preference for those actions that are nearly greedy (have a higher value) or particularly uncertain. It is useful for exploring various actions, but it might also lead the agent to try actions that will not give a good reward. In that sense, it would be better to select among the *nongreedy* actions according to their potential to be optimal, considering both how close their estimates are to being maximal and the uncertainties in those estimates.

One effective way of doing this is to use the UCB algorithm. It is based on the principle called optimism in the face of uncertainty, selecting the best action based on a confidence interval [57]. The confidence interval specifies the interval within which the average reward value of the actions lies. Then, the UCB algorithm selects the action that has the highest upper confidence bound to explore.

Specifically, the upper confidence bound corresponding to the action $a_i \in \mathcal{A}$, denoted $UCB(a_i)$, is given by:

$$UCB(a_i) = \sqrt{\frac{c \ln t}{N_t(a_i)}}, \quad \text{for } i = 0, 1, ..., I - 1, \tag{2-7}$$

where t denotes the current step number, $N_t(a_i)$ the number of times that action a_i has been selected prior to t, and the parameter c ($c > 0$) controls the degree of exploration [2]. Similarly, it is possible to express the action selection rule of the UCB algorithm as follow:

$$A_t = \underset{a_i}{\operatorname{argmax}}[Q_t(a_i) + UCB(a_i)], \quad \text{for } i = 0, 1, ..., I - 1, \tag{2-8}$$

where $Q_t(a_i)$ is the estimate value of action a_i at the time step t, and $UCB(a_i)$ is given by Eq. (2-7) [2].

The idea behind the UCB algorithm is summarized in Algorithm 1. As can be noticed, using this algorithm, all actions will eventually be selected, but actions with lower value estimates, or that have already been chosen frequently, will be selected with decreasing frequency over time. UCB often performs well, but it is more complicated than *ϵ-greedy* and *decreasing ϵ-greedy* to extend beyond the MAB problem (when agents have to learn to behave in more than one situation). Also, it has difficulty in dealing with nonstationary problems, when more complex methods would be needed [2].

Algorithm 1: The UCB algorithm

```
Algorithm parameters:
    Action set: $\mathcal{A} = \{a_0, a_1, ..., a_{I-1}\}$, where $I = |\mathcal{A}|$ is the total number
    of available actions;
    $c > 0$, controls the degree of exploration;
Initialize:
    $N_t(a_i) = 0$, for $i = 0, 1, ..., I - 1$;
    $Q(a_i) = 0$, for $i = 0, 1, ..., I - 1$;

for each episode do
    while $t$ is not the final step do
        if $t < I$ then
            $A_t \leftarrow a_{i=t}$                # Initialy, it tries all the actions
        else
            Choose $A_t$ from $\mathcal{A}$ according to Eq. (2-8)
        end if
        $N_t(A_t) \leftarrow N_t(A_t) + 1$
        Take action $A_t$ and observe $R_t$
        Update $Q(A_t)$ according to Eq. (2-2)
        Update $UCB(a_i)$ according to Eq. (2-7)
    end while
end for
```

2.3.3 Markov Decision Process

Markov Decision Process (MDP) offers a mathematical framework for modeling decision-making situations. It states that any RL problem can be reduced to three signals being exchanged between an agent and its environment: one signal to represent the choices made by the agent (the actions), one signal to represent the basis on which the choices are made (the states) and one signal to represent the agent's goal (the rewards) [2]. Fig 2.3 shows the agent-environment interaction in terms of the mentioned signals.

If the sequence of rewards the agent receives after time step t is denoted $R_{t+1}, R_{t+2}, R_{t+3}, ...$, the agent's goal can be defined as to maximize the *expected return*, where the return, denoted G_t, is defined as some specific function of that reward sequence. In the simplest case, the return corresponds to the sum of the received rewards:

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + ... + R_T, \tag{2-9}$$

where T is the final time step [2].

For *episodic tasks*[5], the expected return can be defined by Eq. (2-9).

[5] The MDP framework defines as *episodic* tasks, those where the agent-environment

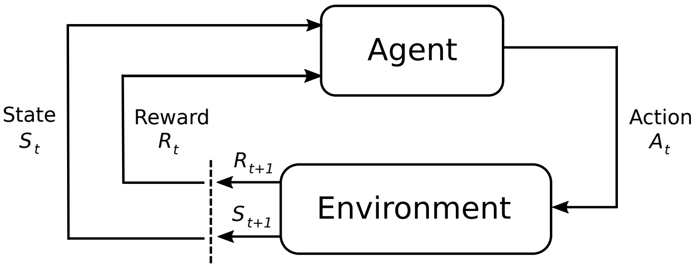

Figure 2.3: The agent-environment interaction in a MDP, where $S_t \in \mathcal{S}$ is the representation of the environment's state received by the agent at each time step t, $A_t \in \mathcal{A}$ is the selected action on the basis of the received state, and $R_t \in \mathcal{R} \subset \mathbb{R}$ is the numerical reward the agent receives as a consequence of its actions. $\mathcal{S}$, $\mathcal{A}$ and $\mathcal{R}$, denote the sets of states, actions and rewards, respectively [2].

However, for *continuous tasks*[6], a different definition for the expected return is required. The reason is that in continuous tasks the final time step would be $T = \infty$, and the expected return, which is what the agent wants to maximize, could be infinite. In this case, it is possible to redefine G_t in terms of a parameter γ, $0 \geq \gamma \leq 1$, which is called the *discount rate*:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}. \tag{2-10}$$

The discount rate controls the importance of immediate and future rewards. If $\gamma = 0$, the agent is "myopic". It is concerned only with maximizing immediate rewards, that is, to learn how to select A_t to maximize only R_{t+1}. As γ approaches 1, the agent becomes more "farsighted" because the expected return takes into account future rewards more strongly.

Another way to express the expected return for continuous tasks is the following:

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + ... \\ &= R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + ...) \\ &= R_{t+1} + \gamma G_{t+1}, \end{aligned} \tag{2-11}$$

which often makes it easy to compute returns from reward sequences.

In section 2.3.1, it was already defined a *policy* as a mapping from states to the probabilities of selecting each possible action. In MDP, if the agent is following the policy π, those probabilities can be denoted as $\pi(a, s)$, were a

interaction is divided into sub-sequences called *episodes*, which ends in a special state called *terminal state*.

[6] In MDP, *continuous* tasks are those that do not have a terminal state, thus will never end.

represents the selected action, and s, the current state. Likewise, for MDPs, a *value function* can be formally defined by:

$$v_\pi(s) = \mathbb{E}[G_t|S_t = s] = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s\right], \text{for all } s \in \mathcal{S}, \tag{2-12}$$

where $\mathbb{E}[.]$ denotes the expected value of a random variable given that the agent follows the policy π, and t is any time step. The function v_π is also called the *state-value function for policy* π [2].

Similarly, the expected return starting from state s, taking the action $a \in \mathcal{A}$, and thereafter following the policy π, which is known as the *action-value function for policy* π, denoted $q_\pi(s, a)$, can be defined as follows [2]:

$$q_\pi(s, a) = \mathbb{E}[G_t|S_t = s, A_t = a] = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s, A_t = a\right]. \tag{2-13}$$

A fundamental property of value functions is given by:

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\Big[r + \gamma v_\pi(s')\Big], \text{for all } s \in \mathcal{S}, \tag{2-14}$$

where $p(s', r|s, a)$ denotes the probability of moving from a state s to another state s', and receiving a reward r by performing the action a. It can be derived from Eq. (2-12) [2] and is known as the *Bellman equation for* v_π.

The Bellman equation (2-14) expresses a relationship between the value of a state and the values of its successor states. It constitutes the basis of different approaches to compute, approximate, and learn v_π, helping to solve the MDP, which actually means finding the optimal policies and value functions.

An *optimal policy* is a policy whose expected return is greater than or equal to that of other policies for all states. In finite[7] MDPs there is always at least one optimal policy, denoted by π_*. All the optimal policies share the same state-value function, called the *optimal state-value function*, denoted v_*, and defined as

$$v_*(s) = \max_\pi v_\pi(s), \text{ for all } s \in \mathcal{S}. \tag{2-15}$$

Optimal policies also share the same *optimal action-value function*, denoted q_*, and defined as follows:

$$q_*(s, a) = \max_\pi q_\pi(s, a), \text{ for all } s \in \mathcal{S} \text{ and } a \in \mathcal{A}(s). \tag{2-16}$$

From Eq. (2-15), it is possible to derive the Bellman equation for v_*, known as the *Bellman optimality equation for* v_* or the *Bellman optimatily*

[7] In a finite MDP, the sets of states, actions and rewards ($\mathcal{S}$, $\mathcal{A}$, and $\mathcal{R}$) all have a finite number of elements [2].

equation:

$$
\begin{aligned}
v_*(s) &= \max_{a \in \mathcal{A}(s)} q_{\pi_*}(s,a) \\
&= \max_a \mathbb{E}_{\pi_*}[G_t | S_t = s, A_t = a] \\
&= \max_a \mathbb{E}_{\pi_*}[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a] \qquad \text{(by (2-11))} \\
&= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] \qquad (2\text{-}17) \\
&= \max_a \sum_{s',r} p(s', r | s, a)[r + \gamma v_*(s')]. \qquad (2\text{-}18)
\end{aligned}
$$

The Bellman optimality equation has a unique solution for finite MDPs. It is actually a system of equations (one for each state) that can be solved using any of the existing methods for solving systems of nonlinear equations. Once v_* has been obtained, it is relatively easy to determine an optimal policy. For each state s, there will be at least one action at which the maximum is obtained in the Bellman optimality equation. Any policy that assigns nonzero probabilities only to those actions is an optimal policy [2].

Although it is possible to find an optimal policy by explicitly solving the Bellman optimality equation, this solution is rarely directly useful. This is mainly because, in practice, there may not be enough computational resources to complete the computation of the solution. Thus, in RL, it is common to use methods that approximately solve the Bellman optimality equation by using currently experienced transitions in place of knowledge of the expected transitions. There are a variety of such methods, including Dynamic Programming (DP), Monte Carlo methods, and Temporal-Difference (TD) Learning [2, 56].

Since TD Learning is one of the central and novel ideas of RL, the rest of this chapter focuses on introducing the main concepts behind it. Also, it will be described Q-Learning, which is one of the most widely used algorithms in TD Learning.

2.3.4 Temporal-Difference learning and the Q-Learning algorithm

TD learning is a model-free learning algorithm that can be understood as a combination of the Monte Carlo and DP ideas. Like the Monte Carlo method, it does not require model dynamics; and like DP, it does not need to wait until the end of the episode to estimate the value function [57].

In TD learning there are two possibilities: *TD Prediction* and *TD Control.* TD prediction focuses on estimating the value function v_π, while TD control aims to optimize it.

TD prediction algorithms use the following rule for updating the value of a state:

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)], \qquad (2\text{-}19)$$

where $\alpha \in (0,1]$ is a small value that controls the learning rate, and is called the *step-size parameter* or *learning rate.*

Looking a little closer at Eq. (2-19), it can be noticed that it is actually the difference between the estimated value of S_t ($V(S_t)$) and the best estimate ($R_{t+1}+\gamma V(S_{t+1})$). This quantity is known as the *TD error*, and can be formally defined as:

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t). \qquad (2\text{-}20)$$

In TD control, there are two kinds of control algorithms: On-policy and Off-policy algorithms. On-policy TD control methods estimate $q_\pi(s,a)$ for the current behavior policy π and for all states s and actions a, while Off-policy methods approximate q_*, the optimal action-value function, independent of the policy being followed [2].

One of the most popular TD control algorithms is Q-Learning [58], which specifies the following rule for updating the values of state-action pairs:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]. \qquad (2\text{-}21)$$

The entire Q-Learning algorithm is presented in Algorithm 2. From Algorithm 2, it is worth highlighting that in Q-Learning, the policy is still important because it determines which state-action pairs are visited and updated (line 4).

Algorithm 2: The Q-Learning algorithm (Off-policy TD Control)

```
Algorithm parameters:
    Action set: $\mathcal{A} = \{a_0, a_1, ..., a_{I-1}\}$, where $I = |\mathcal{A}|$ is the total number
    of available actions;
    step-size: $\alpha \in (0,1]$;
    epsilon: $\epsilon > 0$;
Initialize:
    $Q(s, a_i) = 0$ for all $s \in \mathcal{S}$ and all $a_i \in \mathcal{A}$ $(i = 0, 1, ..., I-1)$;

1: for each episode do
2:     Initialize S
3:     while S is not terminal do
4:         Choose $A$ from $S$ using a policy derived from $Q$ (e.i., $\epsilon$-greedy)
5:         Take action $A$, observe $R$ and $S'$
6:         $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_{a_i} Q(S', a_i) - Q(S,A)]$
7:         $S \leftarrow S'$
8:     end while
9: end for
```

In RL problems with only one state, it is possible to use a simplified version of the standard Q-Learning algorithm. It is known as *stateless* Q-Learning [59, 60] and defines the following rule for updating the values of the actions:

$$Q(A_t) \leftarrow Q(A_t) + \alpha[R_{t+1} - Q(A_t)]. \tag{2-22}$$

From Eq. (2-22) it should be noticed its similarly with Eq. (2-3). The difference here, is that *stateless* Q-Learning, instead of using a variable step-size, it uses a fix step-size (α), which allows giving more weight to recent rewards than to long-past rewards [2].

The RL methods described in this chapter are currently the most widely used in RL [2]. This is probably because of their simplicity: they can be applied in real-time, with a minimal amount of computation, and also, they can be expressed by single equations that can be easily implemented with small computer programs. Both the UCB and stateless Q-Learning algorithms will be used in this thesis when designing the RL algorithms to locally control the transmission parameters of LoRaWAN end devices.

3
Analyzing and Evaluating the Performance of LoRaWAN

This chapter introduces the network architecture of LoRaWAN as well as a detailed description of its network stack, starting from the physical layer and working up in the Open System Interconnection (OSI) model up to the application layer. Furthermore, it presents a performance evaluation of LoRaWAN under different scenarios, demonstrating through simulations LoRaWAN's limitations in terms of reliability and scalability.

3.1
Network architecture and network stack of LoRaWAN

Fig. 3.1 shows the network architecture of LoRaWAN. As a typical LPWAN, a LoRaWAN network consists of nodes, gateways, the network server, and different application servers. Among these elements, one of the most important is the LoRaWAN network server. It is the orchestrator of the network, being responsible for monitoring nodes and gateways, forwarding incoming packets to the corresponding application server, among other functionalities [1].

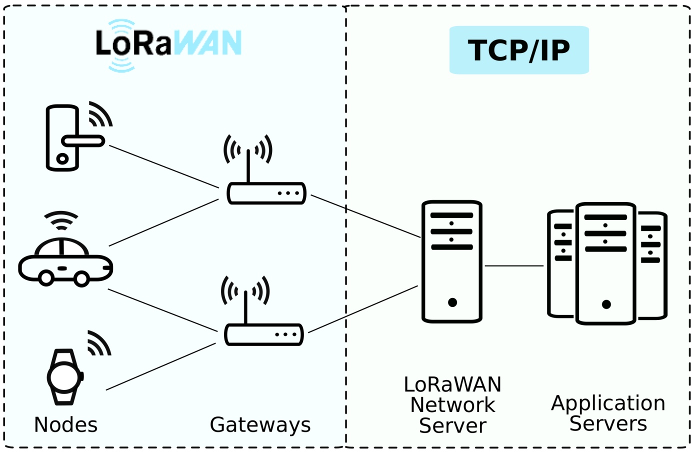

Figure 3.1: Network architecture of LoRaWAN [1].

The remainder of this section extends the introduction to the network stack of LoRaWAN presented in Section 2.1.2.1. It describes the fundamental aspects regarding each layer of LoRaWAN's network stack, paying particular attention to the physical and MAC layers, whose understanding is essential to analyze the performance of LoRaWAN networks and, subsequently, to find new solutions to improve it.

3.1.1 Physical layer

As introduced in Section 2.1.2.1, the physical layer of LoRaWAN consists of LoRa, a flexible modulation scheme that allows multiple bandwidths and configurable data rates, constituting the key to the success of LoRaWAN. Since LoRa is based on the CSS technique, more details about this modulation scheme are presented in Appendix A. The following two sections describe the operation of the physical layer at gateways and nodes, as well as a brief discussion about packet collisions in LoRaWAN based networks.

3.1.1.1 Physical layer for nodes

The physical layer of LoRaWAN nodes can be in one of the following states: transmission (*TX*), reception (*RX*) or idle (*IDLE*) state, which are controlled from the MAC layer [61].

During the reception state, the physical layer keeps track of the instantaneous Signal-to-Noise Ratio (SNR) of each arriving packet so that high powered messages can destroy a colliding message. When a message is successfully received, it is forwarded to the MAC layer for further processing. In the other direction, when a packet is ready for transmission (coming from the MAC layer), the physical layer immediately tries to transmit it, notifying the MAC layer if the transmission fails for any reason. Whenever the physical layer has sent or received its message, it returns to the *IDLE* state [61].

3.1.1.2 Physical layer for gateways

The physical layer for gateways is different from the physical layer for nodes in the sense that gateways allow demodulating several packets with different data rates (SFs) simultaneously[1], even on the same channel [62]. On the other hand, it should be noticed that the uplink and downlink of LoRaWAN occur on the same channels and use the same modulation technique. As a result, LoRaWAN gateways do not have any full-duplex capabilities. That is, the self-interference of a gateway can destroy all messages on that channel on that gateway.

[1] The number of simultaneous demodulation is an arbitrary system parameter that may be set to any value for a customer-specific circuit. This thesis considers gateways that can demodulate up to 8 packets simultaneously, which is a typical value for this parameter [62].

3.1.1.3 Packet collisions

Although LoRa modulation enables decoding multiple messages with different SFs simultaneously, packet collisions can still result in packet losses. Two types of collisions can occur in a LoRaWAN network: (1) collisions between packets with the same SF, or (2) collisions between packets with different SFs [20]. LoRaWAN establishes specific rules for determining when the packets involved in each type of collisions are going to be lost, which are summarized in Fig. 3.2 [3].

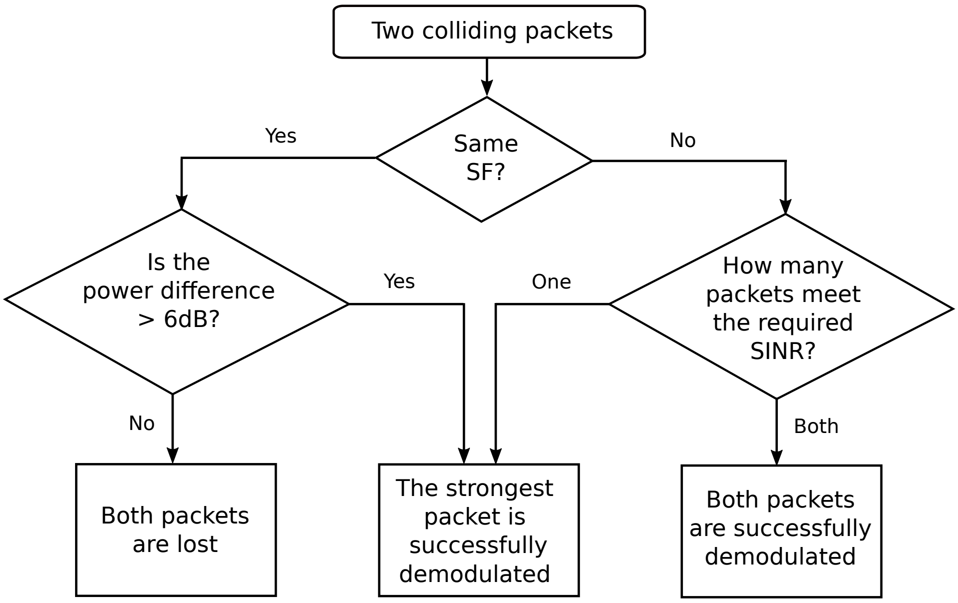
Two colliding packets
Same SF?
Yes
No
Is the power difference > 6dB?
Yes
One
How many packets meet the required SINR?
No
Both
Both packets are lost
The strongest packet is successfully demodulated
Both packets are successfully demodulated

Figure 3.2: LoRaWAN rules for discarding packets when a collision has occurred [3].

As depicted in Fig. 3.2, if the colliding packets have the same SF (data rate), the strongest packet will be successfully demodulated only if the power difference between them is greater than 6dB. Otherwise, both packets are going to be lost. On the other hand, if the colliding packets have different SF, it could be possible to correctly demodulate both of them. In this case, the packet losses are determined by the required SINR of the current SF according to the LoRaWAN specifications [4]. For example, when considering the collision of two packets with SF7 and SF9, which require SINRs of -7dB and -12.5dB, respectively, the packet with SF7 will be successfully demodulated as long as the packet with SF9 is no more than 7dB stronger. Likewise, the packet with SF9 could also be correctly demodulated if the packet with SF7 is no more than 12.5dB stronger.

From Fig. 3.2, it should be noticed that when colliding two packets with the same SF, at least one of them is going to be lost. This is worse than when colliding two packets with different SFs because in this case, at least one packet is going to be correctly demodulated. Furthermore, this figure demonstrates the importance of combining the power and spreading factor control in LoRaWAN networks. Adjusting the SF will decrease the collision probability between packets with the same SF, while the power control will help to avoid packet losses once a collision has occurred, which can greatly improve the network performance in terms of reliability and scalability.

3.1.2 Medium Access Control layer

The MAC protocol operating upon the physical layer described in the previous sections is LoRaWAN. As a MAC protocol, it is responsible for controlling nodes' access to the shared medium, and for implementing mechanisms that efficiently allocate the available resources among them. The following sections describe the operation of the LoRaWAN protocol at both gateways and end devices.

3.1.2.1 MAC layer for nodes

The MAC layer of a LoRaWAN node is very simple. Whenever a new message comes from the application layer, it first checks if there is a free channel on which to transmit the message to comply with the duty cycle regulations. If there is an available channel, the node transmits the message on the channel found, using a specific data rate (SF) and transmit power. If no channel is available, the node tries again after a random time.

Following each successful uplink[2] transmission, the end device opens two short receive windows for potential downlink[3] messages. End devices can not transmit another uplink message before it either has received a downlink message in any of the receive windows corresponding to the previous transmission, or the second receive window has expired [4].

LoRaWAN nodes can send two types of traffic: confirmed or unconfirmed traffic. Confirmed traffic will request the gateway to respond with an ACK during one of the receive windows opened after the packet transmission. On the other hand, both, confirmed and unconfirmed messages, are transmitted

[2] Uplink messages are sent by nodes to the LoRaWAN network server through one or many gateways [4].

[3] Downlink messages are sent by the LoRaWAN network server to only one node through a single gateway. The current LoRaWAN specification does not describe the transmission of multicast messages from a network server to many end devices [4].

NbTrans[4] times, except if a valid downlink message is received after one of the transmissions. If the end device has reached its maximum number of retransmissions without receiving an ACK, it can try to regain connectivity by increasing the SF. It is up to the end device to retransmit the message again or to drop it and move on. Finally, LoRaWAN specification strongly recommends the use of a data rate adaptation procedure when transmitting confirmed traffic. It states that in the absence of the expected ACK after transmitting a confirmed message, the packet retransmission should happen in a lower data rate (higher SF), which corresponds to more robust symbols and a large reach [4].

3.1.2.2 MAC layer for gateways

While end devices transmit their uplink messages using a random access approach, the LoRaWAN gateways use a synchronized downlink. Whenever a gateway receives a new message, it schedules two downlink slots so that at least one second later, it can send its downlink messages in one of the scheduled slots. Received messages are forwarded as a whole to the application layer, and, in the meanwhile, the gateway waits for a response from the network server. If it replies with a message to discard the downlink transmission, the scheduled downlink slots are discarded. On the other hand, when receiving a confirmed data message, the gateway should respond with an ACK using one of the scheduled downlink slots.

3.1.2.3 Adaptive Data Rate mechanism

In Section 3.1.1.3, it was already discussed that managing the transmission parameters plays an important role in determining the reliability and scalability of LoRaWAN networks. To this end, LoRaWAN implements the ADR mechanism, which dynamically modifies the data rate and transmit power at the end devices through the appropriate MAC commands, as depicted in Fig. 3.3. As can be observed, the ADR mechanism has two parts running asynchronously, one at the network server and the other at the end device[5]. When an end device wants the network server to manage its transmission parameters, it will set the ADR bit in its uplink messages. Then, the network server will control the transmission parameters of the end device through the appropriate

[4] This parameter can be adjusted by the network manager to control the redundancy of the node uplinks, in order to achieve the desired Quality of Service (QoS) [4].

[5] This thesis considers the performance evaluation of LoRaWAN networks where only the ADR mechanism at the nodes' side is enabled.

LoRaWAN MAC commands[6]. When the network server is unable to control the data rate of the end device[7], or when the ADR bit is not set in uplink messages, the end device can still manage its transmission parameters itself using the ADR mechanism that resides at the node's side [4].

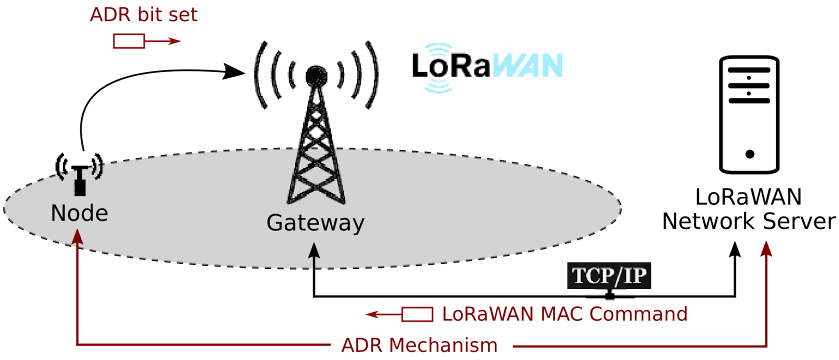

Figure 3.3: Adaptive Data Rate mechanism of LoRaWAN.

Fig. 3.4 represents how the ADR mechanism works at the end device's side, according to the LoRaWAN specifications [4]. The operation of the ADR mechanism is based on a counter (ADR_ACK_CNT) and two configurable parameters (ADR_ACK_LIMIT and ADR_ACK_DELAY). The ADR_ACK_CNT counter is initialized to zero and then increased by one each time a new message is sent, and the default values of the ADR_ACK_LIMIT and ADR_ACK_DELAY parameters is 64 and 32, respectively.

As shown in Fig. 3.4, initially, nodes keep transmitting uplink messages and checking if the ADR_ACK_CNT counter has reached the configured value for ADR_ACK_LIMIT (64 by default). If after ADR_ACK_LIMIT uplink messages ($ADR\_ACK\_CNT >= ADR\_ACK\_LIMIT$) any downlink message has been received, the end device must set the $ADRACKReq$ bit in the subsequent uplink transmission, requesting the network server to respond with a downlink packet within the next ADR_ACK_DELAY uplink messages. Any downlink message received during this interval resets the ADR_ACK_CNT counter. If no reply is received after a total of $ADR\_ACK\_LIMIT + ADR\_ACK\_DELAY$ uplink messages, the end device must try to regain connectivity by adjusting its transmission parameters. As can be observed, the node first steps up the transmit power ("Tx power") to the maximum value, if possible[8]. If increasing the transmit power up to the

[6] The LoRaWAN MAC commands involved in the ADR mechanism can be found in the LoRaWAN specification [4].

[7] ADR control may not be possible when the radio channel attenuation changes fast and constantly [4].

[8] In LoRaWAN, the default transmit power is the maximum transmit power allowed for the device considering its capabilities and regional regulatory constraints [4].

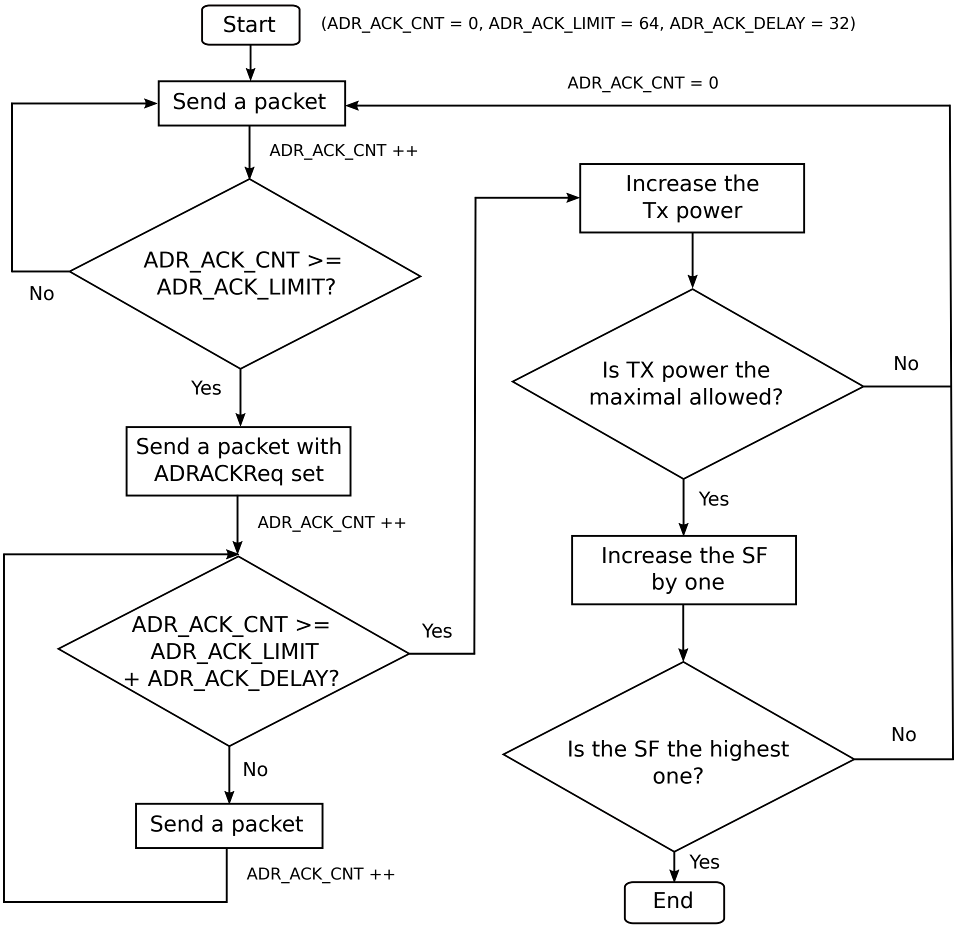
Start
(ADR_ACK_CNT = 0, ADR_ACK_LIMIT = 64, ADR_ACK_DELAY = 32)
Send a packet
ADR_ACK_CNT = 0
ADR_ACK_CNT ++
ADR_ACK_CNT >= ADR_ACK_LIMIT?
No
Yes
Send a packet with ADRACKReq set
ADR_ACK_CNT ++
ADR_ACK_CNT >= ADR_ACK_LIMIT + ADR_ACK_DELAY?
Yes
No
Send a packet
ADR_ACK_CNT ++
Increase the Tx power
Is TX power the maximal allowed?
No
Yes
Increase the SF by one
Is the SF the highest one?
No
Yes
End

Figure 3.4: LoRaWAN ADR mechanism at the node's side, where "Tx power" represents the transmit power.

maximum value is not enough, the end device must further lower its data rate step by step every time *ADR_ACK_DELAY* is reached [4]. The reliability is increased by decreasing the data rate because, as explained in Appendix A, in LoRa modulation, a lower data rate corresponds to a higher SF, which leads to more robust symbols and requires a lower SNR at the receiver to correctly demodulate the signals.

The effectiveness of the ADR mechanism in improving the LoRaWAN's network reliability is going to be assessed in Section 3.2.

3.1.3 Transport and Application layers

The transport and application layers reside just above the MAC layer in the network stack of LoRaWAN (Fig. 2.1). It should be noticed that there is no networking layer in the LoRaWAN's network stack since there is no routing involved in LoRaWAN networks.

3.1.3.1
Application layer for nodes

The application layer for nodes could be any IoT application, whose generated data should be delivered to a particular application server through the LoRaWAN gateways. These IoT applications have specific requirements in terms of data rate, network delay, among others, which the MAC layer tries to accomplish.

3.1.3.2
Application layer for gateways and the network server

In a LoRaWAN network, the functionalities of the application layer are split. One part is implemented at the gateway and the other at the network server. At gateways, the application layer works as a translation application. It collects the messages received by the LoRaWAN MAC layer, encapsulates them in TCP/IP packets, and sends them to the remote network server, and vice versa. On the other hand, the application layer implemented at the network server is considered the brain of the LoRaWAN network. It decides which nodes are part of the network and which nodes are not, determines which gateway is responsible for talking to each node, and removes duplicated messages. It is the layer where the ADR mechanism operates at the network server's side, and it is also the layer for implementing new algorithms that control the transmission parameters of the end devices in a centralized manner.

3.2
Performance evaluation of LoRaWAN

This section focuses on investigating through simulations the performance of LoRaWAN under various scenarios and considering important metrics such as the average PER, network throughput, fairness, average network delay, and average energy consumption.

3.2.1
LoRaWAN simulator

In the past few years, several simulation tools have been proposed for evaluating the performance of LoRaWAN [29]. One of the most well-known, earliest, and simple LoRaWAN simulator is LoRaSim [63]. It is open-source, based on Python, and gives great insights into the LoRaWAN performance. However, LoRaSim does not implement MAC commands, or ACKs (does not allow confirmed traffic) or any downlink traffic possibility. Thus, it can not be used to analyze the network performance when nodes adjust their transmission parameters based on the feedback from the gateway or when using the ADR mechanism. Although the work in [64] extends LoRaSim to support downlink traffic, it still does not allow ACKs or MAC commands. Similarly, a LoRaWAN simulator based on Omnet++ has been proposed in [65]. It implements the ADR scheme and allows multiple gateway deployments; however, it does not include MAC commands.

The Network Simulator 3 (NS-3) [66] is one of the most widely used simulators to evaluate the performance of LoRaWAN [29]. Several LoRaWAN NS-3 modules can be found in literature [17, 25, 67], which allow studying LoRaWAN's performance under certain heavy condition scenarios, such as a large number of nodes or high traffic loads. The LoRaWAN module proposed by authors in [67], allows multiple-gateway deployments and features MAC commands. However, it does not allow confirmed traffic (does not implement ACKs) and ignores the self-interference existing in LoRaWAN networks due to the half-duplex capabilities of the gateways. Besides, in their implementation, there is no TCP/IP based network for the communication between the network server and gateways. This absence of the TCP/IP network might become important because bottlenecks and high latencies can also affect network performance. On the other hand, the authors in [25] proposed an NS-3 module for LoRaWAN that also supports multiple gateways and confirmed traffic. Still, they did not include MAC commands, and, in their implementation, the network server controls the individual gateways directly without using a TCP/IP network.

By comparing with the above LoRaWAN NS-3 modules, a comprehensive solution has been proposed by authors in [17]. This implementation allows multiple gateways, confirmed traffic, and includes MAC commands. It is in accordance with the LoRaWAN v1.0 class A specification and offers a flexible backbone architecture that allows the easy integration of new protocols. Gateways and the network server are connected through a TCP/IP network, which allows simulating bottlenecks or other network drawbacks. From the programming point of view, it also provides base C++ classes that will enable easy implementation and integration of new algorithms and MAC commands [17].

Table 3.1 summarizes the main features of the LoRaWAN simulators that have been discussed through this section. As can be noticed, among the existing solutions, the LoRaWAN NS-3 module proposed in [17] is the most complete. It is also highly configurable and allows evaluating the impact of different parameters on LoRaWAN's performance. Furthermore, it enables the easy integration and evaluation of new algorithms and protocols for improving the scalability and reliability of LoRaWAN [17]. Due to the above, and because it is publicly available[9], the LoRaWAN NS-3 module proposed in [17] was chosen as the simulation tool for assessing and analyzing LoRaWAN's performance in this thesis.

Table 3.1: LoRaWAN simulators comparison.

Proposal/ Publication	Environment	Multiple gateways	Downlink Traffic	Confirmed Traffic	MAC Commands
LoRaSim/ [63]	Python	Yes	No	No	No
LoRaWANSim/ [64]	Python	Yes	Yes	No	No
FLoRa/ [65]	Omnet++	Yes	Yes	Yes	No
NS-3 module/ [67]	NS-3	Yes	Yes	No	Yes
NS-3 module/ [25]	NS-3	Yes	Yes	Yes	No
NS-3 module/ [17]	NS-3	Yes	Yes	Yes	Yes

Finally, it is worth highlighting that despite its interesting features, the NS-3 based LoRaWAN simulator used in this thesis, has some small implementation flows that differ from the LoRaWAN specification. Although these variations are not going to significantly affect the performance evaluation of LoRaWAN, they are listed below:

- The end devices should wait for a random time (in the range between one and three seconds) before transmissions, which does not really happen in LoRaWAN.
- At gateways, if any uplink traffic is being received during the first downlink window, the transmission of potential downlink messages is delayed to the second downlink slot.

3.2.2 System model and simulation setup

All the analysis and simulation results presented in this section consider class A end devices following the European regulations for LoRaWAN [4]. In that sense, the spectrum is limited to the default allowed channels in Europe, corresponding to 868.1, 868.3, and 868.5 MHz, each enabling data rates from DR0 (SF12) to DR5 (SF7) and bandwidth of 125kHz. Likewise, the data rate and transmit power adaptation at the end devices is carried out through the ADR mechanism described in Section 3.1.2.3.

[9] https://github.com/networkedsystems/lora-ns3

On the other hand, it is important to clarify that under confirmed traffic scenarios, the SF will be increased by one after two lost ACKs to improve reliability. Likewise, the transmitted packet will be discarded if the maximum number of retransmissions ($NbTrans$) is reached without receiving the expected ACK. The configurable parameters of LoRaWAN ($NbTrans$ and the number of simultaneous receptions at gateways) and the ADR mechanism (ADR_ACK_LIMIT and ADR_ACK_DELAY) are set according to Table 3.2, as well as the initial amount of energy for the nodes' batteries. Finally, the Okumura-Hata and Rayleigh models are combined for modeling the propagation losses due to the distance and multipath fading, respectively.

Table 3.2: Simulation parameters.

Parameter/Unit	**Scenario 1**	**Scenario 2**	**Scenario 3**
Number of gateways/-	1	7	1
Number of simultaneous receptions at gateways/-	8	8	8
Number of nodes/-	100, 500, and 1000	100, 500, and 1000	100, 500, and 1000
NbTrans/-	1	1	8
ADR_ACK_LIMIT/-	64	64	64
ADR_ACK_DELAY/-	32	32	32
Packet length (payload)/bytes	51	51	51
Average inter-packet interval/seconds	120	120	120
Simulation time/hours	24	24	24
Initial energy/Joules	100	100	100

The considered scenarios vary depending on the type of traffic (confirmed or unconfirmed) and the number of gateways (single gateway or multiple gateways). In single gateway scenarios, only one gateway is enabled. It is placed at the center of the coverage area, which corresponds to the dark-blue region shown in Fig. 3.5 (a circle of 1000 meters radius). For multiple gateway scenarios, seven gateways are deployed, as depicted in Fig. 3.5. In this case, the coverage area is the light-blue circle of 1500 meters radius, and the distance between gateways is 1000 meters. Nodes are uniformly[10] distributed over the coverage area, which, according to the above, depends on the scenario used. Both nodes and gateways maintain a fixed position during the entire simulation time, but nodes are at the height of one meter above the ground, while the gateways at 30 meters.

[10] To guarantee the uniform distribution of the nodes over a circular area, was used the

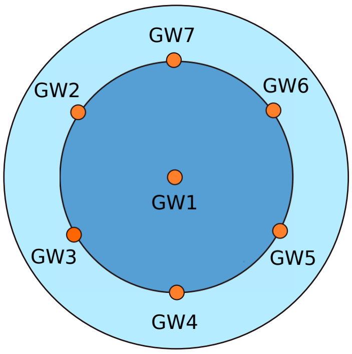

Figure 3.5: Coverage area and gateways' location for single gateway and multiple gateways scenarios.

Finally, the nodes' application layer generates messages of 51 bytes every 120 seconds, which are sent to the network server through the enabled gateways. The simulation creates different numbers of nodes for each evaluation (100, 500, or 1000 nodes), and the simulation time corresponds to a full day of 24 hours in all the cases. The evaluation of a particular scenario considers ten iterations. That is, each scenario is simulated ten times and, the final results, correspond to the average results over these iterations.[11]

3.2.3 Analysis scenarios and considered metrics

The following scenarios were formulated to evaluate LoRaWAN's scalability and reliability:

- **Scenario 1:** single gateway and unconfirmed traffic. It is a simple network topology where nodes transmit unconfirmed messages to the network server. Thus, nodes do not request or wait for ACKs.
- **Scenario 2:** multiple gateways and unconfirmed traffic. It is a variation of Scenario 1 where seven gateways are enabled. It will allow analyzing the spatial diversity of LoRaWAN since multiple gateways can respond to a particular node in the network.
- **Scenario 3:** single gateway and confirmed traffic. It is also a variation of Scenario 1 where all nodes transmit confirmed traffic, requesting the network server for ACKs. It was formulated to evaluate the impact of confirmed traffic on network reliability since ACKs provide a guarantee of delivery.

UniformDiscPositionAllocator Class offered by the NS-3 simulator.

[11] The same seed is used for all the evaluations (the default seed of the NS-3 simulator). That is, each time that a particular scenario is evaluated, the same results are obtained.

All the simulation parameters associated with the scenarios described above are summarized in Table 3.2. Below, a brief explanation of the considered metrics to evaluate the performance of LoRaWAN.

1. *PER:* The PER is used to analyze the reliability and scalability of LoRaWAN. It is calculated as follows:

$$PER = 1 - \frac{N_{Rx(APP)}}{N_{Tx(APP)}}, \tag{3-1}$$

   where $N_{Rx(APP)}$ is the number of received packets at the network server's application layer, and $N_{Tx(APP)}$ is the number of generated packets by the node's application layer, which were delivered to the MAC layer for transmission. It should be noticed that this approach contemplates the packet losses because of collisions or link unreliability in the LoRa based network, but also the packet losses in the TCP/IP network due to congestion or other network drawbacks.

2. *Network throughput:* It corresponds to the overall throughput and is calculated as the overall PDR multiplied by the network packet rate [68]. The overall PDR is given by:

$$PDR = \frac{N_{Rx(MAC)}}{N_{Tx(MAC)}}, \tag{3-2}$$

   where $N_{Rx(MAC)}$ corresponds to the total number of packets received by the gateway's MAC layer (from all nodes in the network), and $N_{Tx(MAC)}$ is the total number of transmitted packets by all the nodes in the network, considering retransmissions. Likewise, the network packet rate is the total number of packets that are sent in the whole network by a unit of time.

3. *Fairness:* This metric evaluates the network fairness in terms of the radio resource allocation. It is measured according to the Jain Index (JI) [69] as follows:

$$JI(x_1, ..., x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \times \sum_{i=1}^{n} x_i^2}, \tag{3-3}$$

   where x_i corresponds to the performance metric (i.e., PDR[12] or throughput) achieved by the end device i. It should be noticed that a JI equal to '1' means a perfectly fair network in terms of the considered metric. Likewise, the lower is the JI, the more unfair the network behaves [68].

[12] In this thesis the Jain Index calculation is based on the PDR.

4. *Average network delay:* When considering confirmed traffic, the average network delay corresponds to the average time elapsed since a packet transmission until the reception of the corresponding ACK. For unconfirmed traffic, it is estimated as the average time elapsed since the transmission of an uplink message, until the reception of any downlink message from the gateway.

5. *Average energy consumption:* In this thesis, the energy consumption is measured in Joules (J) and calculated as the average amount of energy consumed by all nodes in the network during the simulation time.

The next sections present the simulation results obtained when evaluating the performance of LoRaWAN in NS-3. They provide a detailed description of the network behavior under different scenarios, and a performance comparison in terms of the metrics described above.

3.2.4 Single gateway and unconfirmed traffic (Scenario 1)

Fig. 3.6 shows the simulation results for Scenario 1, where all nodes try to communicate with gateway GW1 and send unconfirmed traffic. This figure represents the average PER as a function of the distance from the central gateway and considering different numbers of nodes. Therefore, it allows arriving at conclusions about the reliability and scalability of LoRaWAN under the mentioned conditions.

As can be observed in Fig. 3.6, for all the cases, nodes close to the gateway experienced better reliability compared to nodes that are further away. It demonstrates how the capture effect impacts the performance of LoRaWAN networks: the closer to the gateway, the higher the received power and thus the lower PER, because high powered packets survive collisions, while low powered packets get lost, as explained in Section 3.1.1.3. Likewise, it should be noticed that the larger the network, the worse performance. The reason is that in dense networks, the probability of packet collisions would be higher. Besides, according to the ADR mechanism (explained in Section 3.1.2.3), nodes increase their SF to improve reliability only after 96 uplink messages without receiving any response from the network server. In that way, all nodes use the SF7 almost all the time, which corresponds to the less robust symbols and the higher SNR required at the gateway to correctly demodulate signals.

According to the above, one can think that a potential solution to improve network performance in LoRaWAN is to set lower values for the parameters ADR_ACK_LIMIT and ADR_ACK_DELAY. By doing this, the ADR

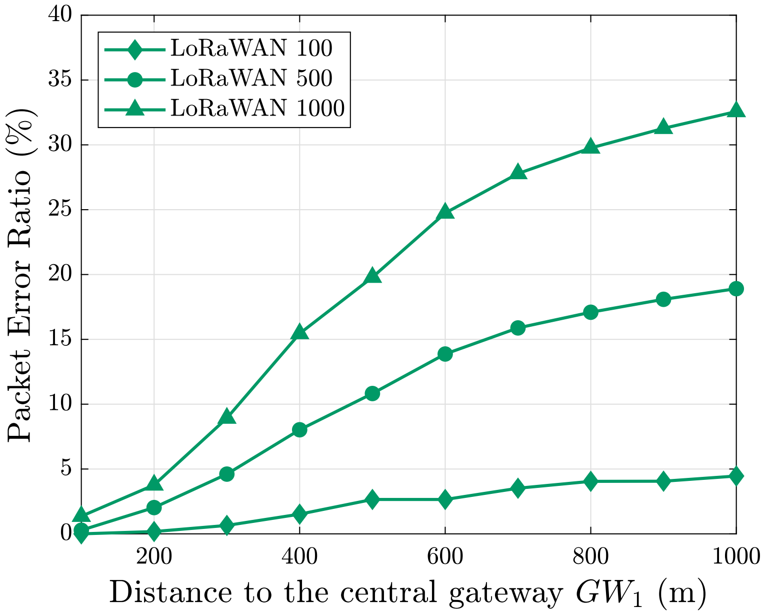

Figure 3.6: Average PER versus distance to the central gateway GW_1 (Scenario 1). At the figure's legend, *"LoRaWAN N"* indicates that this curve corresponds to a LoRaWAN network with N nodes.

mechanism will increase the SF of the end devices earlier, trying to regain connectivity with the gateway to improve reliability. However, this approach will not succeed. The reason is that all the nodes will use higher SFs earlier, which would only increase the probability of packet collisions due to a higher SF leads to a larger symbol time, as explained in Appendix A. In that way, a better solution is to take advantage of the LoRa modulation and implement algorithms that efficiently distribute the available SFs among the nodes in the network. It will decrease the collision probability between packets with the same SF and, consequently, the packet losses.

3.2.5 Multiple gateways and unconfirmed traffic (Scenario 2)

Fig. 3.7 illustrates the simulation results corresponding to a LoRaWAN network with seven gateways (as depicted in Fig 3.5), where nodes send unconfirmed traffic to the network server through all of them (Scenario 2).

By comparing the simulation results shown in Fig. 3.6 and Fig. 3.7, it can be noticed that the reliability under multiple gateways scenarios is considerably higher than under single gateway deployments. Specifically, in the worse case (1000 nodes), Scenario 2 shows a 28% lower PER than Scenario 1. The reason behind these improvements is as follows. Under Scenario 2, when two packets collide at a specific gateway, it is unlikely that both senders (nodes) are exactly at the same distance from him. Therefore, that gateway

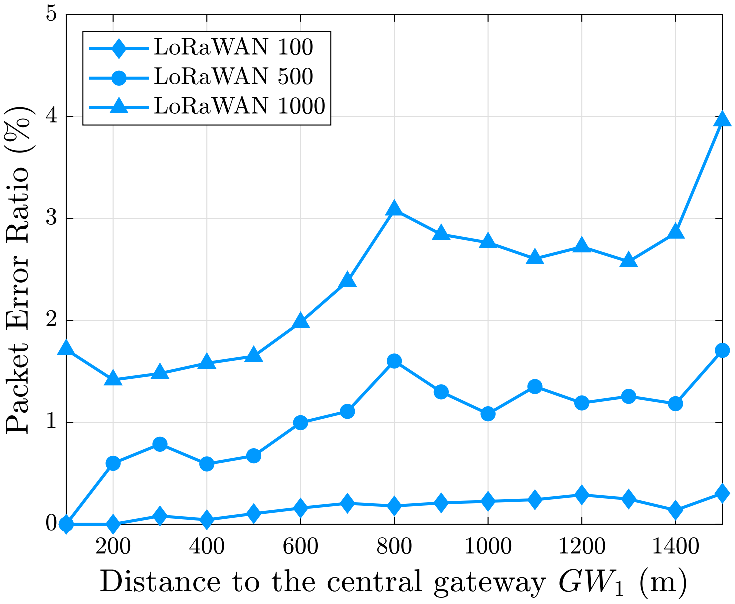

Figure 3.7: Average PER versus distance to the central gateway GW_1 (Scenario 2). At the figure's legend, *"LoRaWAN N"* indicates that this curve corresponds to a LoRaWAN network with N nodes.

can successfully demodulate the most robust packet but could drop the weaker one. As there are more gateways in the network, the potential dropped packet will be correctly demodulated by a neighboring gateway, because the path losses in the other direction make it stronger than the other message.

On the other hand, from Fig. 3.7, it should be noticed that the best PER values are experienced by nodes that are at less than 500 meters from GW_1, or around 1000 meters away from him. That is, when nodes are close to GW_1 or near gateways GW_2, ..., and GW_7.

From the results above, it is possible to conclude that multiple gateways are an attractive solution to achieve a better performance in LoRaWAN networks. However, it would increase the cost of the network infrastructure, which could be a limitation in most cases.

3.2.6 Single gateway and confirmed traffic (Scenario 3)

Fig. 3.8 presents the simulation results corresponding to Scenario 3, a LoRaWAN network where nodes send confirmed messages, expecting an ACK for each of them (GW_1).

As depicted in Fig. 3.8, for small networks, the PER is considerably low when transmitting confirmed traffic. Specifically, for a network with 100 nodes, it is zero for all distances from the central gateway. However, the PER rapidly increases when considering larger networks. As can be observed, for networks

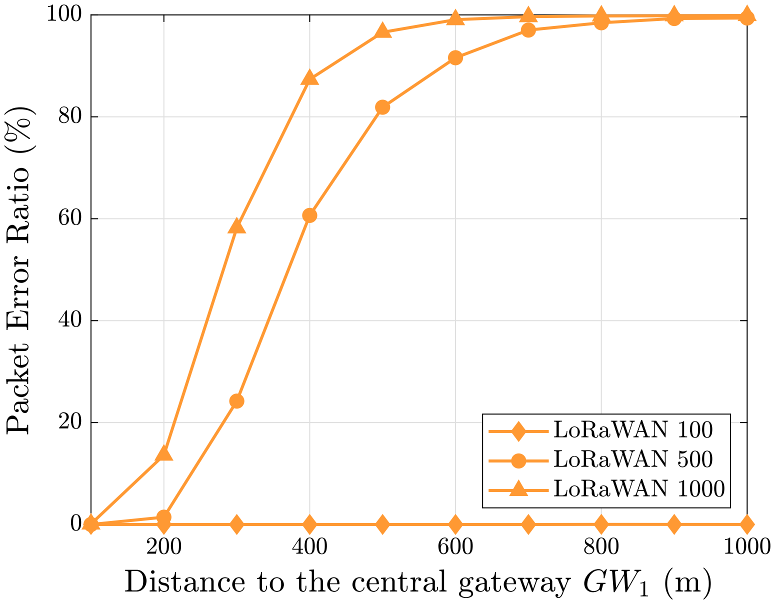

Figure 3.8: Average PER versus distance to the central gateway GW_1 (Scenario 3). At the figure's legend, *"LoRaWAN N"* indicates that this curve corresponds to a LoRaWAN network with N nodes.

with 500 and 1000 nodes, the PER experienced by nodes far from the central gateway is almost 100%. The reasons for these results are as follows.

In Section 3.1.2.1 it was already explained that confirmed messages are retransmitted $NbTrans$ times (8 times in this case), except if the corresponding ACK is received. Likewise, due to the adopted mechanism to improve reliability when transmitting confirmed traffic, the SF will be increased by one after two lost ACKs. These extra transmissions (the exact number of retransmissions is the difference between the red and orange bars in Fig. 3.9), especially transmissions with high SFs, cause more collisions and increase the probability of packet losses. More importantly, nodes far from the central gateway get stuck in a vicious circle: they switch to a higher SF to get a more reliable channel, but doing this, they only increase the probability of more collisions.

On the other hand, in Section 3.1.1.2, it was explained that LoRaWAN gateways do not have any full-duplex capabilities, which means that uplink transmissions collide with downlink messages. Therefore, the ACK transmissions will interfere with the reception of uplink data messages, and this interference increases with the number of nodes in the network. The simulation results in Fig. 3.8 demonstrate this, showing excellent performance for small networks and extremely poor reliability when increasing the number of nodes.

To conclude, one can say that transmitting confirmed messages could be a solution to improve reliability in small LoRaWAN networks. However, it should be noticed that there is a trade-off between reliability and energy

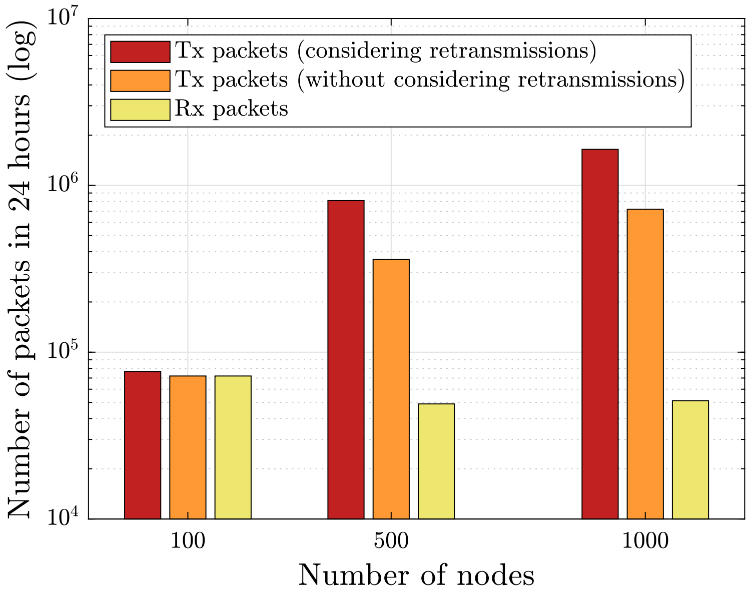

Figure 3.9: The total amount of transmitted (Tx) and received (Rx) packets in 24 hours (Scenario 3). Red bars count all the transmissions at the MAC layer (considering retransmissions of the same packet), orange bars represent the total number of packets that arrive at the MAC layer from the application layer for transmission, and yellow bars score the total number of successfully received packets at the gateway's MAC layer.

efficiency because packet retransmissions and ACK receptions will increase the energy consumption of the end devices. Likewise, it was shown that ACKs in LoRaWAN do not scale, the network reliability is extremely low when considering a large number of nodes transmitting confirmed traffic.

3.2.7 Performance comparison

This section presents a performance comparison between the scenarios described in the previous sections (Scenarios 1, 2, and 3). It considers important metrics that represent the requirements of many IoT applications, such as the average PER, average network delay, network throughput, fairness, and average energy consumption.

Average Packet Error Ratio

Fig. 3.10 summarizes the simulation results discussed in the three former sections. It clearly shows that, on average, small LoRaWAN networks (100 nodes) can provide very low PERs, being less than 3% for the simplest LoRaWAN setup: single gateway and unconfirmed traffic (Scenario 1). Even better reliability can be achieved under small networks through either more dense gateway deployments (Scenario 2) or sending confirmed traffic (Scenario 3). However, in both cases, the improvements in reliability have a price that should be paid. For multiple gateway deployments, a more expensive network, and when transmitting confirmed traffic, higher energy consumption and average network delay, as will be shown in short.

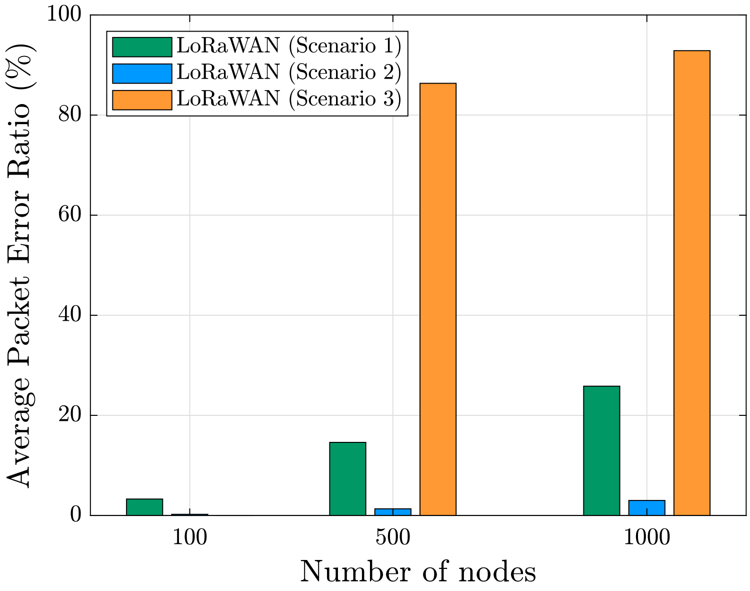

Figure 3.10: Average PER under the scenarios 1, 2, and 3.

On the other hand, Fig. 3.10 shows that in larger networks (500 and 1000 nodes), the better reliability is achieved under scenarios with multiple gateways and unconfirmed traffic (Scenario 2). Besides, it illustrates how, in LoRaWAN, confirmed traffic is not scalable. It offers an average PER of 0% for networks with 100 nodes, but more than 80% for networks with 500 and 1000 nodes.

Average network delay

Fig. 3.11 shows the average network delay under scenarios 1, 2, and 3. As can be observed, there is a notable increase in the average network delay when transmitting confirmed messages (Scenario 3), especially for larger networks. The reason is that in large networks, many packets are retransmitted (See Fig. 3.9), thus delaying the reception of the corresponding ACKs.

From Fig. 3.11, it should also be noticed that the average network delay under Scenario 3 with 500 nodes, is much higher than that obtained delay for 1000 nodes. This is due to how this metric is calculated. It was defined in

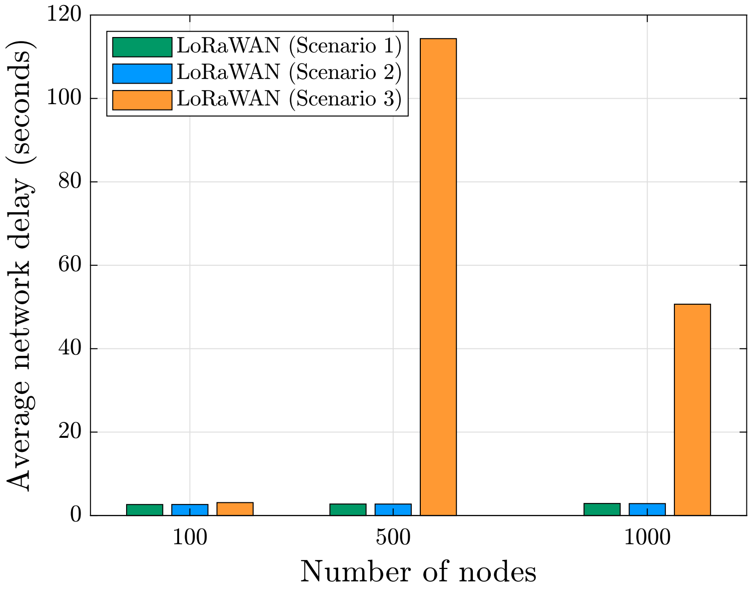

Figure 3.11: Average network delay under the scenarios 1, 2, and 3.

Section 3.2.3 as the average time elapsed since a packet transmission until the reception of the corresponding ACK; thus, its computation ignores discarded packets. Fig. 3.10 already showed that in large LoRaWAN networks with confirmed traffic, almost all the packets are discarded when the maximum number of retransmissions is achieved without receiving the expected ACK. Therefore, with 1000 nodes, the probability of having high delay values when calculating the average network delay for each node, is lower than when considering 500 nodes.

Network throughput

Fig. 3.12 presents the network throughout under the studied scenarios. As can be observed, the best throughput corresponds to deployments with multiple gateways and unconfirmed traffic (Scenario 2). By comparing the results in Fig. 3.10 and Fig. 3.12, one can notice that the best PER does not necessarily correspond to the best throughput. For example, Scenario 3 with 100 nodes offers the best achievable PER (0%), but also the worst throughput. This is because the throughout calculation is based on the PDR, which is a MAC layer metric involving packet retransmissions. In contrast, the PER was defined in this thesis as an application layer metric that does not take into account the packet retransmissions at the MAC layer. Therefore, in this case, minimizing the PER and maximizing the throughput is not the same.

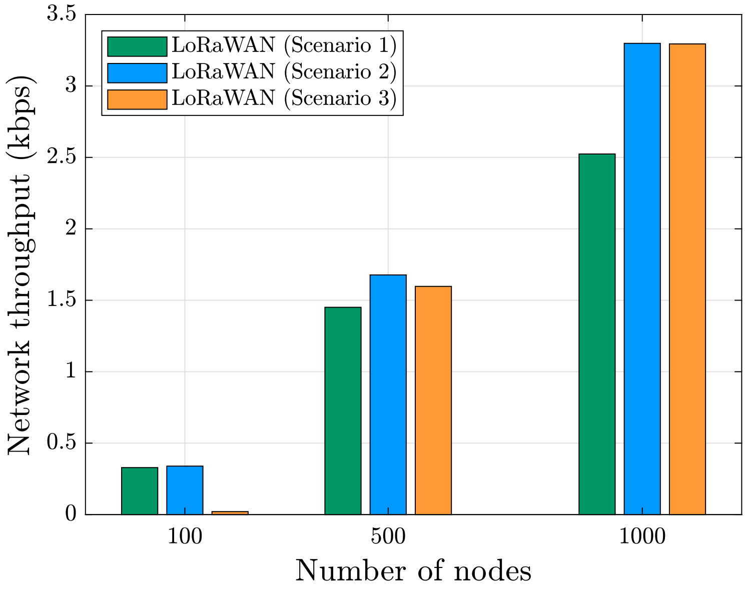

Figure 3.12: Network throughput under the scenarios 1, 2, and 3.

Fairness

Fig. 3.13 exhibits the fairness under the scenarios 1, 2, and 3. As can be seen, the highest fairness is achieved through multiple gateway scenarios and unconfirmed traffic (Scenario 2). Furthermore, this figure shows that Scenario 3 results in very unfair networks when increasing the number of nodes. The reasons have already been explained in Section 3.2.6. Due to the adopted mechanism for adjusting the SFs when transmitting confirmed traffic, nodes far from the central gateway experienced a higher number of packets losses than close nodes, and this effect increases with the number of end devices in the network.

Average energy consumption

Finally, Fig. 3.14 illustrates the average energy consumption under scenarios 1, 2, and 3 when varying the number of nodes in the network. It shows that, on average, the amount of energy consumed by a LoRaWAN node is around 3 Joules in 24 hours. Furthermore, it should be noticed the increase in the average energy consumption when transmitting confirmed traffic (Scenario 3). The reason is that confirmed messages are retransmitted *NbTrans* times if the expected ACK is not received. This extra transmission, and the energy spent on receiving ACKs, explains the higher energy composition in Scenario 3 compared with the scenarios 1 and 2.

On the other hand, by comparing Fig. 3.10 and Fig. 3.14, it can be demonstrated the trade-off between energy consumption and reliability under

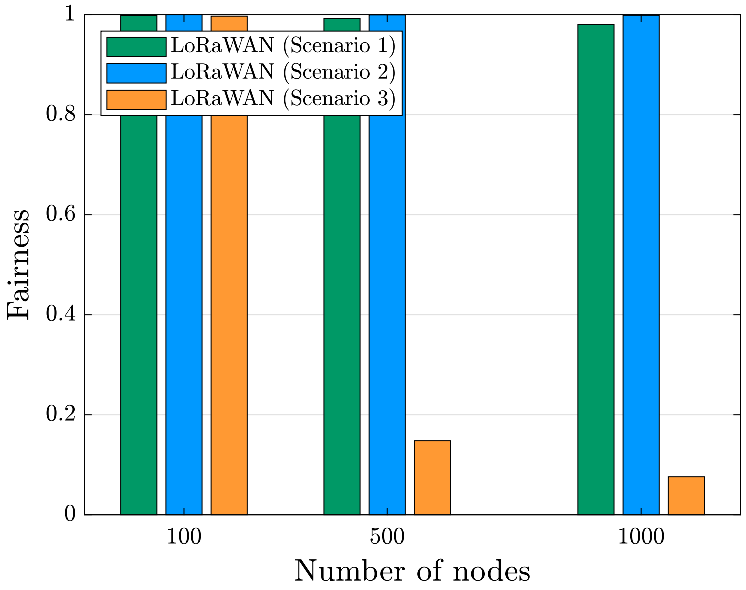

Figure 3.13: Fairness under the scenarios 1, 2, and 3.

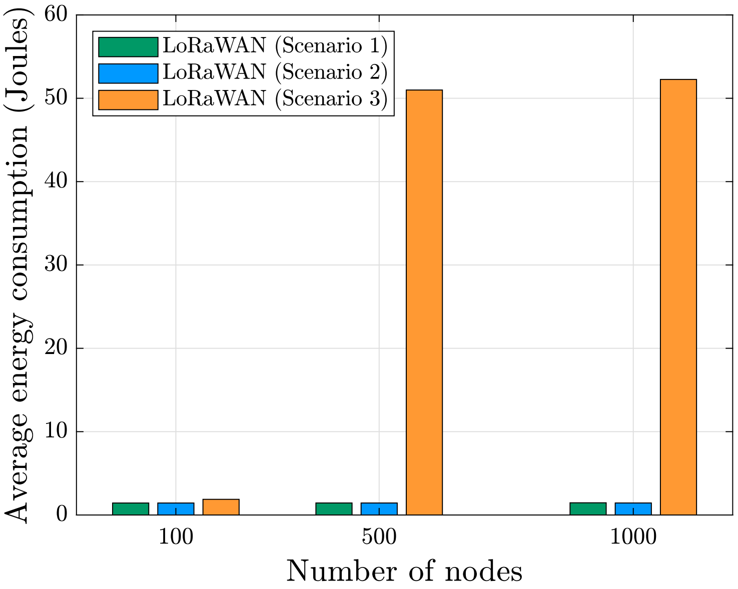

Figure 3.14: Average energy consumption under the scenarios 1, 2, and 3.

Scenario 3. These figures show that, when considering 100 nodes, there is a PER of 0%; however, there is also a higher energy consumption when compared with the other scenarios.

To conclude, this chapter has investigated LoRaWAN's performance under three different scenarios, by varying the number of enabled gateways (single gateway and multiples gateways) and the type of traffic (confirmed and unconfirmed traffic). The simulation results demonstrated that it is challenging to guarantee reliability in LoRaWAN. Adding more gateways looks like the best option in this sense; however, it will increase the costs, which could be a limitation. On the other hand, transmitting confirmed traffic can provide excellent reliability for small networks (100 nodes) at the expense of higher energy consumption and network delay, as well as lower throughput. Furthermore, it was shown that requesting an ACK for each message does not work for large networks (500 and 1000 nodes), when, on average, the 80% of the transmitted packets are lost.

At this point, some interesting questions could be the following:

- Is it possible to provide more reliability without deploying additional gateways, but only leveraging the SS modulation scheme of LoRaWAN?
- How to allow many nodes in the network while still providing a good PER and range? That is, how to make LoRaWAN networks scalable?

The next chapter answers these questions. It will introduce different algorithms based on RL that dynamically adjust the SFs and transmit power at the end devices, leading to significant improvements of LoRaWAN's performance in terms of reliability and scalability.

4
Improving the LoRaWAN MAC Layer Through Reinforcement Learning Techniques

The previous chapter has exposed the limitations of LoRaWAN and, in particular, has shown that connecting a massive number of LoRaWAN nodes creates severe challenges for network scalability, which is even more challenging due to the capture effect. It also showed that, although LoRaWAN offers a mechanism for adapting the SFs and transmit power at the end devices (the ADR mechanism), it is not enough to achieve reliable communications under dense networks. Despite this, it should be noticed that LoRaWAN still offers opportunities for improvements. The fact that LoRa modulation allows multiple data rates and transmit powers, enables the development of new algorithms to improve network performance by efficiently allocating these transmission parameters.

4.1
Challenges and related work

Recently, several research works have demonstrated that properly assigning SFs and transmit powers can improve communication in LoRaWAN [20,31–34]. More specifically, these investigations have proved that SFs are the key to affect LoRaWAN's achievable reliability and scalability. However, selecting the proper set of SFs is a difficult task involving many issues. First, a good SF for a particular node depends on the distance from that node to the gateway. In some cases, in order to reach the gateway, a node can only transmit using SF12 (the highest SF allowed for LoRaWAN). On the other hand, if nodes close to the gateway use SF12, they would unnecessarily occupy the channel for a long time because, as explained in Appendix A, it corresponds to the highest symbol time. More importantly, the set of SFs used by all nodes in the network also matters. As discussed in Chapter 3 (Section 3.1.1.3), when many nodes use the same SF, packet collisions are inevitable. In such cases, it could be appropriate to employ higher SFs because they require lower SINRs to correctly demodulate the signals.

To date, most of the proposals for distributing SFs in LoRaWAN networks do it statically. Although it can improve network performance by decreasing the collision probability between packets with the same SF, some weaknesses can be associated with the static assignation of SFs. One issue is that these approaches imply communication overhead since the SF allocation should be distributed, and the other one is that once a communication channel is blocked, end devices can not recover from this. An interesting work addressing the mentioned issues was presented by authors in [37]. They proposed RS-LoRa, a new MAC protocol based on LoRaWAN, where nodes can locally select their transmission parameters by using some scheduling information that is dynamically provided by gateways through beacon messages. Their simulation results show that RS-LoRa can improve LoRaWAN's reliability and scalability, demonstrating the advantages of adjusting the transmission parameters during run-time. It is an expected result because their approach allows nodes to adapt the SF and transmit power according to the network dynamics: the number of nodes can increase/decrease, interfering signals can appear/disappear, and the end devices' traffic patterns can change [37].

According to the above, using online mechanisms for dynamically adjusting the transmission parameters at the end devices looks like a prominent approach to optimize the performance of LoRaWAN networks. In this sense, RL algorithms could be an excellent tool for designing such online mechanisms. This is because controlling the transmission parameters at each end device can be seen as a decision-making task where nodes have to learn the best set of parameters for transmitting their data, and, as mentioned in Chapter 2, RL algorithms are ideal for solving this kind of tasks.

In the literature, two research works applying RL techniques for locally controlling the transmission parameters of LoRaWAN nodes have been identified [42, 43]. Both of them are distributed learning approaches where each node implements a MAB algorithm, namely, UCB or EXP3 [70], for adapting its transmission parameters when considering stochastic or adversarial environments, respectively.[1] Their simulation results show significant improvements in terms of the probability of successful data transmission and energy consumption. However, there are some limitations related to their proposals and how they are evaluated. The first issue is that both systems rely on ACKs for rewarding the actions selected by the RL algorithms running at nodes, which is very challenging in real implementations due to the high packet losses occurring in LoRaWAN networks when using confirmed traffic (demonstrated in Section 3.2.6). This drawback is not reflected in their simulation results be-

[1] In the LPWANs context, the environment should be modeled as adversarial in case of considering external interference. That is, when the interference generated by other technologies operating on the same frequency band can interrupt the feedback messages (the rewards).

cause their simulator implements a perfect downlink channel, where ACKs are never lost, and the ACK transmissions do not interfere with the data packet receptions at the gateway. On the other hand, their evaluations only consider single gateway scenarios, and in [43], only 100 nodes are enabled for evaluating the proposed algorithms. Finally, they do not tune the parameters of the used RL algorithms, i.e., the exploration rate c of the UCB algorithm, and they do not present any analysis about the convergence of the proposed learning approaches.

From the above, it is clear that designing RL algorithms for locally adjusting the transmission parameters of LoRaWAN end devices is very challenging. The main reason is the limited feedback that LoRaWAN offers for rating the actions selected by the RL algorithms running at the nodes' side. From Section 3.2.6, one can conclude that using ACKs as reward signals is infeasible in large LoRaWAN networks (500 and 1000 nodes). Besides, the use of ACKs to rate the decisions of the RL algorithms brings an additional drawback from the learning point of view. This is because the RL algorithms could erroneously interpret the no reception of an ACK as a fail in the previous packet transmission (assigning a bad reward to the previously selected action) when it could really correspond to the loss of the expected ACK. In this way, implementing RL algorithms at the nodes' side requires efficient methods for carrying the reward information from gateways to the end devices. That is, without affecting the network performance by adding extra interference and ensuring the rewards reception at the end devices to do not affect the learning process.

To tackle the challenges described through this section, and to address the limitations of the current research works improving the LoRaWAN's performance, this thesis proposes Reinforcement Learning LoRa (RL-LoRa), a novel MAC layer that significantly improves LoRaWAN's reliability and scalability by applying RL techniques. This new protocol involves two key innovations: (1) the use of RL algorithms to locally controlling the transmission parameters of LoRaWAN end devices and (2) a mechanism to provide such algorithms of the required feedback without compromising the network performance. The system design of RL-LoRa is based on RS-LoRa [37] in the sense that uses beacon messages for carrying the reward information from gateways to the RL algorithms that are running at nodes. This approach ensures a lower network interference compared to the use of ACKs as reward signals and, it allows realizing the full potential of the RL algorithms for dynamically adjusting the transmission parameters of LoRaWAN end devices.

The remainder of this chapter presents a detailed description of the proposed RL-LoRa MAC protocol and the implemented RL algorithms to adjust the transmission parameters at the end devices. Likewise, it evaluates the proposed system and algorithms under different scenarios.

4.2 RL-LoRa: system design

RL-LoRa is a *synchronous and distributed approach* to improve LoRaWAN's performance by locally adjusting the transmission parameters (SF and power) of the end devices through RL techniques. It was designed to operate upon the LoRaWAN MAC layer and has two parts running synchronously, one at the gateways and the other at the end devices. Gateways have two main functions: generating and keeping updated the reward information corresponding to each active node in the network, and distributing this information through periodically broadcasting beacon messages. Likewise, in RL-LoRa, beacons have two primary purposes: carrying the reward information from gateways to end devices, and synchronizing the nodes' transmissions. This is because the end devices can transmit their packets only during specific periods (*frames*) that are opened after the reception of a beacon message from the gateway. Then, each LoRaWAN node employs the beacon information to update the parameters of the RL algorithm and, using this latter, selects the transmission parameters for any potential packet transmission. The next sections present the system model of RL-LoRa, offering a detailed description of how gateways and nodes address their main functionalities.

4.2.1 General system description

As depicted in Fig. 4.1, in RL-LoRa, the whole available bandwidth is divided into one *synchronous downlink channel* (for gateways to transmit the beacon messages) and K *asynchronous uplink/downlink channels*. All the asynchronous channels are structured in the unit of *frames*, each having the same structure and duration of T_{frame} seconds. Each frame starts with the reception of a beacon, followed by K uplink/downlink slots (one on each available asynchronous channel) during which nodes can transmit their data messages in an ALOHA manner, as they would do in the legacy LoRaWAN protocol [37].

At gateways, a frame counter is used to maintain the current frame number (f), which is included at the beacon header to keep nodes aware of the timing in the network. Each beacon carries the reward information corresponding to all the nodes in the network. It is up to each node to identify

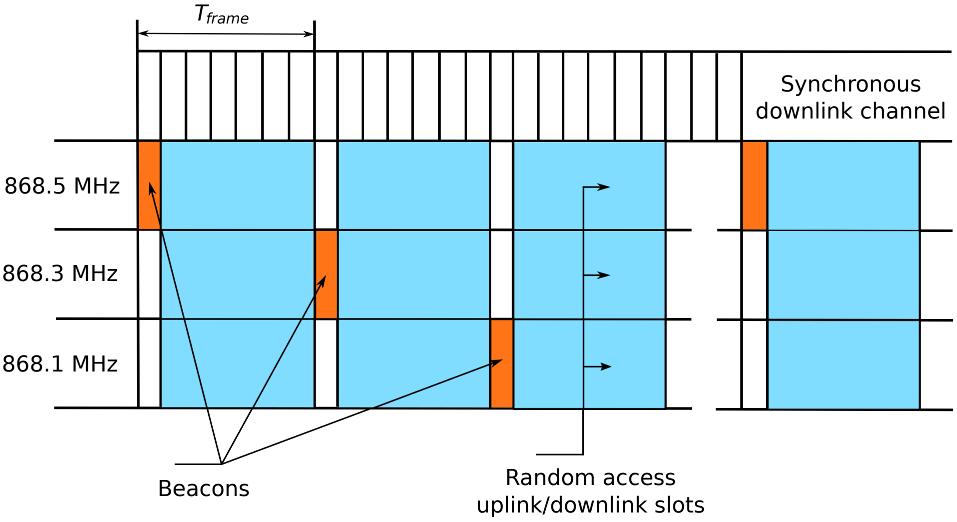
T_{frame}
Synchronous downlink channel
868.5 MHz
868.3 MHz
868.1 MHz
Beacons
Random access uplink/downlink slots

Figure 4.1: Channel assignment in RL-LoRa.

its own reward from the received beacon, which can be done by using the node's address, as will be explained in Section 4.2.3. The beacon sent at the beginning of the frame *f* is carrying the reward information corresponding to the frame *f-1*. That is, the beacon starting the frame *f*, carries information about the reception or not at the gateway of the packets transmitted by all nodes in the network during the frame *f-1*. This information will be used by the RL algorithms running at nodes to keep their parameters updated. Finally, as in LoRaWAN, gateways can choose whether or not to acknowledge the successfully received data messages.

It should be noticed that, in RL-LoRa, although the asynchronous uplink/downlink channels are framed, the access in each frame is still ALOHA. It enables low power consumption since nodes can sleep for a long time [37]. Specifically, during each frame, nodes should wake up only twice: at the frame's beginning to receive beacons (every T_{frame} seconds) and at the selected time for transmission within the current frame (only if they have a new packet to transmit).

4.2.2 Gateways

As mentioned before, in RL-LoRa, the main function of gateways is to generate the reward information corresponding to each active node in the network, and periodically broadcast it through beacon messages.

Beacons transmission

At gateways, each new beacon transmission occurs on a different channel, iterating over all the available channels, as depicted in Fig. 4.1. It will help to avoid collisions with beacons from other gateways and the interference from other networks operating at the same frequency [37]. On the other hand, to ensure the required SNR for correctly receiving beacons at all the nodes in the network (even at those nodes that are far from the gateway), all beacons are transmitted using the maximal transmit power allowed by LoRaWAN (14 dBm). Although, in this case, all beacons are transmitted using the SF9, different SFs could be used, higher or smaller ones, to increase the reception probability of beacons or to avoid exceeding the channel access time, respectively.

Beacons structure

The RL-LoRa beacons use the message format defined in the LoRaWAN specification for downlink traffic [4], which is shown in Fig. 4.2. Specifically, RL-LoRa beacons carry the fields represented in Table 4.1, in the MACPayload field shown in Fig. 4.2.

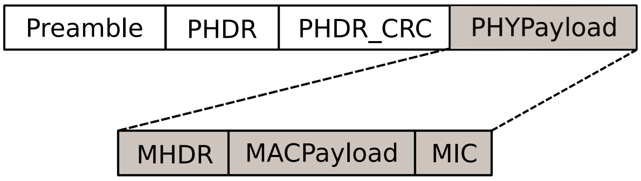

Figure 4.2: The message format of an RL-LoRa beacon, where PHDR is the LoRa physical header (PHDR), PHDR_CRC is the latter's Cyclic Redundancy Check (CRC), and PHYPayload is the physical payload. Inside the PHYPayload field, MHDR represents the MAC header, MACPayload is the MAC payload (it caries the RL-LoRa beacon fields), and MIC is a 4-bytes Message Integrity Code (MIC) [4].

Table 4.1: Fields included in the payload of an RL-LoRa beacon.

Fields	GatewayID	FrameID	NbNodes	RewardInfo
Bits	0:15	16:23	24:31	32:32+N

As depicted in Table 4.1, the first field in the MAC payload of an RL-LoRa beacon is *GatewayID*, which identifies the gateway sending the beacon, enabling nodes to synchronize to different gateways. Next, the *FrameID* field represents the current frame number and gives nodes information about the timing in the network (this value is unique for each beacon sent). The field

NbNodes[2], represents the total number of active nodes, and *RewardInfo* carries the reward information corresponding to all the nodes in the network.

Finally, it should be noticed that the total length of the MACPayload field of an RL-LoRa beacon will depend on the number of active nodes in the network. For example, for networks with 100, 500, or 1000 enabled nodes, it will be 17, 67, or 130 bytes, respectively[3], which is in accordance with the maximum payload size established by the LoRaWAN specification [4]. More details about the generation of the *RewardInfo* field are presented below.

Reward information

In RL-LoRa, the reward information corresponding to all nodes in the network is generated at the beginning of each frame. More specifically, the reward information corresponding to a node n, generated at the beginning of the frame f, indicates if the potential uplink message sent by that node during the previous frame $(f-1)$ was received or not at the gateway. The reward values used by RL-LoRa are '0' and '1', which are assigned according to the following rule:

$$R_{n,f} = \begin{cases} 1, & \text{if the packet transmitted by node } n \text{ during} \\ & \text{the frame } f-1, \text{ was successfully received} \\ & \text{at the gateway.} \\ 0, & \text{otherwise,} \end{cases} \tag{4-1}$$

where $R_{n,f}$ denotes the reward information generated at the beginning of the frame f that corresponds to the node n.

It should be noticed that the *RewardInfo* field carries the reward information corresponding to all nodes in the network. To achieve it, each bit of this field will store the reward information corresponding to a single node. In this way, the length of *RewardInfo* will depend on the number of active nodes in the LoRaWAN network. That is, if there are N enabled nodes, the length of *RewardInfo* will be N bits. It is up to the gateway to generate the *RewardInfo* field, which is done as follows.

[2] This field carries the total number of actives nodes divided by 100. That is, the value of this field will be '1' (decimal format) if the total number of enabled nodes in the network is 100. By doing this, it is possible to reduce beacons' length, and it would be enough since eight bits can represent networks with up to 25600 active nodes.

[3] In case the length of an RL-LoRa beacon does not correspond to an integer number of bytes, the last byte of the beacon frame must be filled with zeros to ensure it.

Gateways should maintain a table with the association between the address[4] of each active node in the LoRaWAN network and its corresponding reward information (generated according to Eq. (4-1)). Such a table should be updated with each beacon transmission and must be sorted in increasing order according to the devices' address. Then, the *RewardInfo* field can be easily generated by concatenating the reward information of each active node ($R_{n,f}$), as shown in Table 4.2.

Table 4.2: The *RewardInfo* field of RL-LoRa beacons.

Bits of the *RewardInfo* field	0	1	...	N
Nodes' reward information for frame f	$R_{0,f}$	$R_{1,f}$	...	$R_{N,f}$
Nodes' address (decimal format)	0	1	...	N

4.2.3 Nodes

As explained in Chapter 3 (Section 3.1.2.3), LoRaWAN networks use the ADR mechanism to adjust the SFs and transmit power at the end devices. By using it, nodes initially transmit their data employing the lowest available SF (SF7), because it corresponds to the shortest symbol time, and thus, minimizes their energy consumption. However, it was already shown in Chapter 3, that this approach can not provide reliable communications when considering large LoRaWAN networks because the probability of packet collision increases significantly when all nodes use the same SF.

In RL-LoRa, the transmission parameters adaptation at the end devices is carried out through the RL algorithms that is running at each node. These algorithms will learn the optimal set of transmission parameters for each end device by interacting with the LoRaWAN network. They take actions that correspond to a combination of SF and transmit power and then send their packets using the selected transmission parameters. Finally, the RL algorithms can learn about the quality of the selected actions by a reward signal that is received through beacons. The next section describes how the task of adapting the transmission parameters at the LoRaWAN nodes, can be formally modeled as an RL problem.

[4] RL-LoRa considers the LoRaWAN nodes' activation via Activation By Personalization (ABP), which allows the network manager to assign arbitrary addresses (DevAddr) to the end devices [4]. For RL-LoRa, it is mandatory assigning to the first active node in the network, the address corresponding to the decimal number '0', and so on. It is necessary for allowing each node to identify its reward information form the received beacons by just using its DevAddr address.

4.2.3.1
Modeling the transmission parameters adaptation at the LoRaWAN nodes as an RL problem

In a LoRaWAN network, the SF and transmit power adaptation at the end devices can be modelled as a Multi-Agent Reinforcement Learning (MARL) problem, where multiple Independent Learners (IL)[5] simultaneously apply RL in a shared environment, as depicted in Fig. 4.3. Specifically, this thesis uses the most basic case of MARL where agents need to learn a strategy for a single state (stateless environment), and the learning challenges stem only from the interaction with other agents. This simplified setting can be seen as a *distributed bandit problem*, and it is known in the literature as an *n-player repeated game* [59, 71].

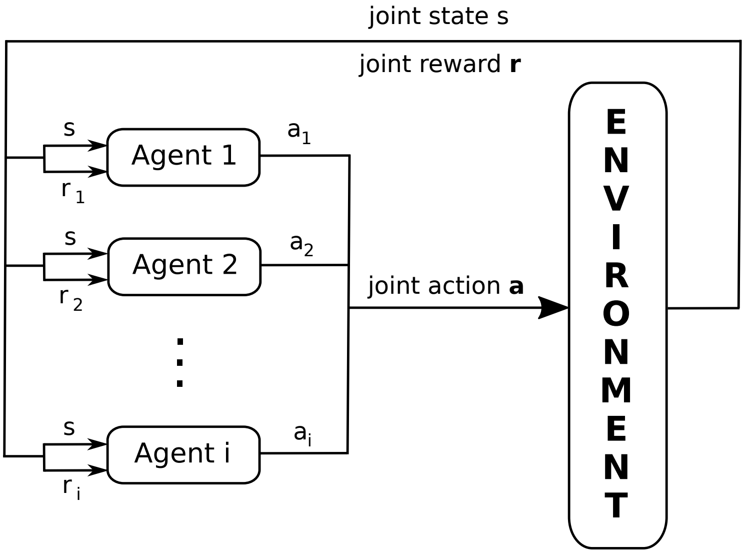

Figure 4.3: Multi-Agent Reinforcement Learning representation.

More formally, it is assumed a collection $\mathcal{N}$ of n agents, and each agent $i \in \mathcal{N}$, has available to it a finite set of individual actions $\mathcal{A}_i$. Agents repeatedly play a stage game in which they independently select an individual action to perform. For each selected action $a \in \mathcal{A}$, agents expect to receive the reward $R(a)$. Finally, agents wish to select actions that maximize the expected reward over the long run.

Although it is clear that selecting actions is a challenging task when considering a multiple-agent setting, agents can still use single-agent RL algorithms to interact in a game environment [59]. This is because, if an agent is unaware of the existence of other agents, can not identify their actions,

[5] When considering Independent Learners, each agent applies RL in the classic sense. That is, each RL agent ignores the existence of other agents, and learns based only on its own reward observation.

or has no reason to believe that other agents are acting strategically, using single-agent RL algorithms is still an appropriate method of learning [71].

A detailed description of the RL agent implemented at each LoRaWAN node, is introduced in the next section.

4.2.3.2 The RL agent running at each node

Chapter 2 introduced the fundamental concepts and elements of RL, and described how a single RL agent could learn the optimal behavior through trial-and-error interactions with its environment. It also described various RL techniques and algorithms which allow an agent to optimize its behavior in a wide range of circumstances, highlighting the UCB and Q-Learning algorithms as two of the most widely applied RL algorithms to solve problems related to communication networks.

This thesis proposes three RL algorithms based on UCB and Q-Learning to locally control the transmission parameters of the LoRaWAN end devices, which will be described in Section 4.2.4. Below, some common definitions to all the proposed algorithms are introduced.

Actions

In terms of the considered sets of actions, two types of RL algorithms were implemented: algorithms for controlling only the SFs, and algorithms for adjusting both SFs and transmit power. The set of considered actions in each case is shown in Table 4.3. As can be observed, when adjusting only SFs, six actions are considered ($a_0 - a_5$), each corresponding with the allowed SFs for LoRaWAN. In this case, all packets are transmitted using the maximal transmit power allowed by LoRaWAN: 14dBm. On the other hand, when controlling both SF and transmit power, twelve actions can be used ($a_0 - a_{11}$), which were defined by combining the allowed SFs for LoRaWAN with two of the available transmit powers (14 and 11 dBm).

Table 4.3: The considered sets of actions by the RL algorithms.

Actions/ Parameters	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}
SF	7	8	9	10	11	12	-	-	-	-	-	-
SF/Power (dBm)	7/14	7/11	8/14	8/11	9/14	9/11	10/14	10/11	11/14	11/11	12/14	12/11

It should be noticed that, although LoRaWAN offers more possibilities for controlling the transmit power (2, 5, 8, 11, and 14 dBm), only two of the available power levels were considered when defining the sets of actions. The reason is that there is a trade-off between the number of actions and the convergence time of an RL algorithm. Although a larger set of actions (considering more transmit power levels) could improve the performance in the long run, it also would increase the learning time of the RL algorithm (the required time to find the best actions).

Environment

In MARL, all the independent agents will be interacting with the same environment. In this sense, since the rewards are clearly not independent and identically distributed (*i.i.d.*) because they depend on the behavior of other agents, the environment can not be considered as a stationary and stochastic environment [72]. Despite this, in this chapter, it will be shown that running a stochastic MAB algorithm (i.e., UCB) at each node is still surprisingly successful.

Rewards

All the proposed RL algorithms in this thesis use the same concept of reward signal. It is given by Eq. (4-1) and is received at each end device through the beacon messages. Since, in RL-LoRa, a single beacon is carrying the reward information of all the active nodes in the network, each end device should identify its own reward from the *RewardInfo* field at the received beacon messages. According to the procedure used by gateways for generating this field (explained in Section 4.2.2 (Reward information)), nodes must use their device address (DevAddr) to extract their reward from it. More specifically, the reward signal associated to a node whose device address is ‘0’ (decimal format), it is given by the value stored at the bit ‘0’ of the *RewardInfo* field.

As explained in Section 4.2.2 (Reward information), the reward signal received by a node n, at the beginning of the frame f, denoted $R_{n,f}$, will be ‘1’ if the potential packet transmitted by that node in the previous frame $(f - 1)$ was successfully received at the gateway, and will be ‘0’ otherwise. According to this rule for generating the reward signals, it should be noticed a node will receive a reward of ‘0’ during the frame f, either if it did not send a data message during the previous frame $(f - 1)$, or if the reception of the potential transmitted message failed. In this way, in RL-LoRa, it is up to the nodes to decide if they should process or not the reward information received by beacons. It is part of the procedure followed by the RL-LoRa nodes when receiving a beacon, which is explained in the next section.

4.2.3.3 Operation of the RL-LoRa nodes

The proposed RL-LoRa protocol states that, at the MAC layer, nodes first must wait for a beacon message. In case of multiple gateways, initially, each node should estimate the closest gateway (based on the RSSs of the received beacons within a pre-defined window) and synchronize its communication with him. Subsequently, the node must listen to all beacons of the gateway it is synchronized with, and execute the following procedure each time a new beacon is successfully received.

Procedure executed by nodes after a successful beacon reception

The procedure executed by an RL-LoRa node after the successful reception of a beacon message is represented in Fig. 4.4. As can be observed, the end devices will parse the information carried by beacons only if there was a packet transmission during the previous frame. This is to prevent the RL algorithm in use from erroneously updating their action values. After deciding if the beacon information will be analyzed or not, the node should check if there is data waiting to be transmitted. If there is no packet to transmit, it will sleep until the next beacon reception, which happens every T_{frame} seconds, at the beginning of each frame. Otherwise, if it is the first packet transmission, the end device will set a specific instant within the frame (Tx_time), which will be used for all its future data transmissions. Then, according to the RL algorithm in use, the node will select the transmission parameters for the current packet and will schedule its transmission for the Tx_time instant within the current frame.

Finally, the end device will wake up at the scheduled time and will intermediately start the packet transmission with the parameters previously determined by the RL algorithm in use. Next to the packet transmission, two receiving slots are reserved, similar to LoRaWAN. The end device will listen to these slots for a potential downlink message from the gateway.

The remainder of this chapter presents the proposed RL algorithms for adjusting the transmission parameters at the end devices, as well as the performance evaluation of RL-LoRa under different scenarios when using such algorithms. It is worth highlighting that the proposed RL-LoRa MAC protocol, as well as the RL algorithms that will be presented in the next sections, were implemented in NS-3. It allows comparing them with the legacy LoRaWAN and RS-LoRa protocols, whose NS-3 implementations are publicly available.

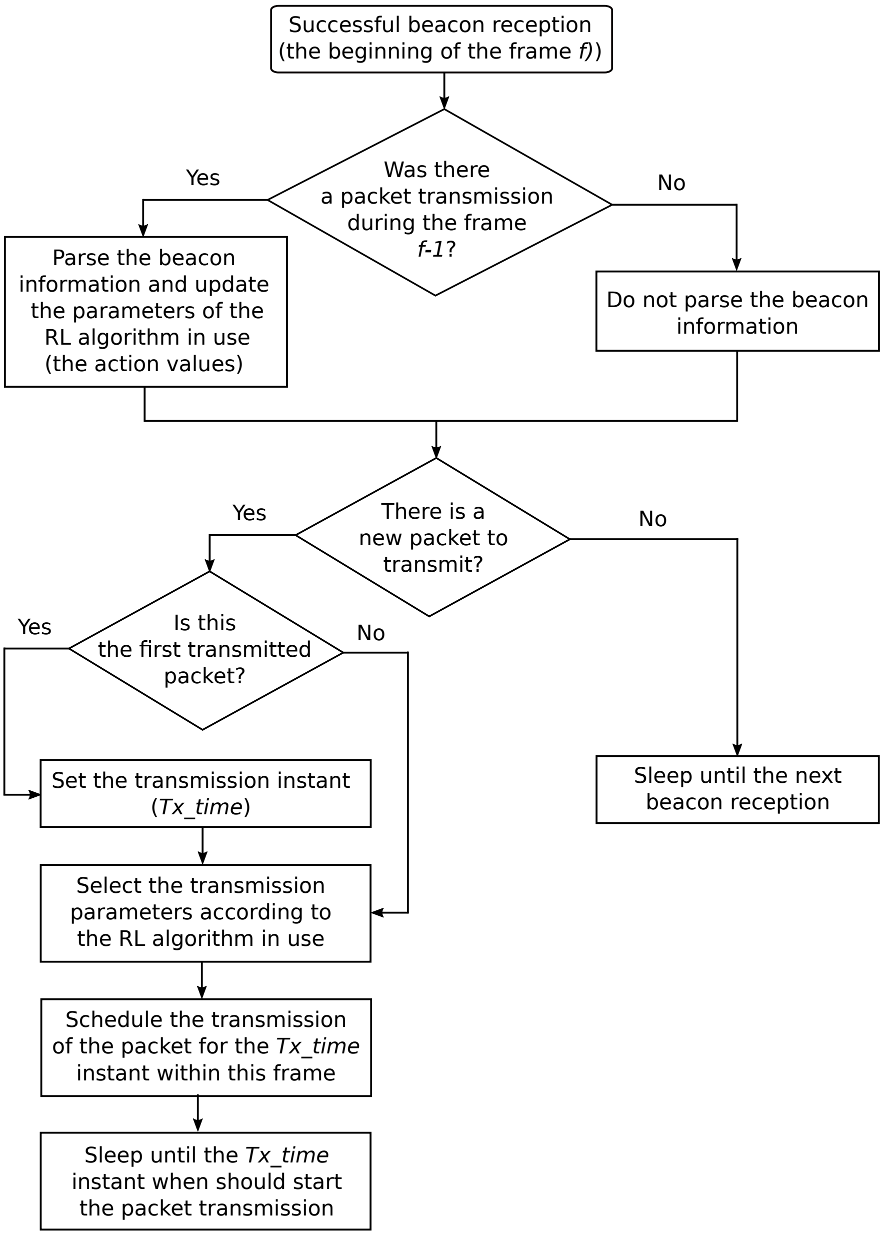
Successful beacon reception (the beginning of the frame *f*))
Was there a packet transmission during the frame *f-1*?
Yes
No
Parse the beacon information and update the parameters of the RL algorithm in use (the action values)
Do not parse the beacon information
There is a new packet to transmit?
Yes
No
Is this the first transmitted packet?
Yes
No
Set the transmission instant (*Tx_time*)
Sleep until the next beacon reception
Select the transmission parameters according to the RL algorithm in use
Schedule the transmission of the packet for the *Tx_time* instant within this frame
Sleep until the *Tx_time* instant when should start the packet transmission

Figure 4.4: Procedure executed by nodes after a successful beacon reception.

4.2.4 Proposed RL algorithms

Three different RL algorithms are proposed in this thesis to work under the RL-LoRa protocol in the task of controlling the transmission parameters at LoRaWAN nodes. They are based on the UCB and stateless Q-Learning algorithms introduced in Chapter 2, and will be broadly described in what follows.

4.2.4.1 The RL-LoRa-UCB algorithm

The first proposed RL algorithm was derived from the UCB algorithm described in Section 2.3.2.3. It is called *RL-LoRa-UCB* and is represented in Algorithm 3. *RL-LoRa-UCB* follows the general operation of an RL-LoRa node (presented in Section 4.2.3.3, Fig. 4.4), but uses the UCB's approach for selecting among the available actions, and the *sample-average* method (given by Eq. (2-3)) for updating the actions values when receiving a beacon message.

As can be observed in Algorithm 3, the *RL-LoRa-UCB* uses the following parameters:

- The set of actions $\mathcal{A} = \{a_0, a_1, ..., a_{I-1}\}$, where each $a_i \in \mathcal{A}$ represents a specific combination of transmission parameters. It will depend on the transmission parameters controlled by the *RL-LoRa-UCB* algorithm (only SFs, or both SFs and transmit power), and it is in accordance with the Table 4.3.
- The frame duration T_{frame}: It specifies the duration of each frame, which also corresponds to the time between successive beacons. It is kept fixed during the whole algorithm's operation time.
- Two arrays containing I variables each one: $Q(a_i)$ and $N(a_i)$ for $i = 0, 1, ..., I-1$, which store the estimate value of each action $a_i \in \mathcal{A}$, and the number of times that these actions has been used, respectively.
- The flags Tx_prev_frame and Pkt_to_Tx, indicating if there was a transmission during the previous frame and if there is a packet to transmit on the current frame, respectively.
- The variable Tx_time, which stores the specific transmission instant for this node within each frame.
- The exploration rate c, which controls the degree of exploration.

At the beginning of the *RL-LoRa-UCB* algorithm, the previously described parameters are initialized, as shown in Algorithm 3.

From Algorithm 3, it should be noticed that the *RL-LoRa-UCB*'s operation is very simple. It will be executed by each RL-LoRa node at the beginning of each frame f and starts by receiving a beacon message. The information carried by that beacon (the reward signal corresponding to that node) will be collected as described in Section 4.2.3.3, only if there was a packet transmission in the previous frame (lines 3-5).[6] Otherwise, the beacon

[6] It should be noticed that the reward information received during the frame f corresponds to the action selected in the previous frame (A_{f-1}).

Algorithm 3: The *RL-LoRa-UCB* algorithm

```
Algorithm parameters:
    Action set: $\mathcal{A} = \{a_0, a_1, ..., a_{I-1}\}$, where $I = |\mathcal{A}|$ is the total number of available actions;
    $T_{frame}$: frame duration (time inter beacon)
    $Q(a_i)$: actions value estimate, for $i = 0, 1, ..., I-1$;
    $N(a_i)$: actions usage counter, for $i = 0, 1, ..., I-1$;
    $A_f$: action selected during the frame $f$;
    $R_f$: reward corresponding to $A_f$;
    $Tx\_prev\_frame$: indicates if there was a packet transmission during the previous frame
    $Pkt\_to\_Tx$: indicates  if there is a packet to transmit on the current frame
    $Tx\_time$: specifies the transmission instant for this node
    $c > 0$: exploration rate;
Initialize:
    $Q(a_i) = N(a_i) = 0$, for $i = 0, 1, ..., I-1$;
    $Tx\_prev\_frame$ = False;

for each frame $f$ do
    Receive the beacon message
    if $Tx\_prev\_frame = True$ then
        $Q(A_{f-1}) \leftarrow Q(A_{f-1}) + \frac{1}{N(A_{f-1})}[R_{f-1} - Q(A_{f-1})]$
    end if
    if $Pkt\_to\_Tx = True$ then
        if $f = 0$ then
            $Tx\_time \leftarrow$ rand(0,$T_{frame}$)   # Select, at random, a value for $Tx\_time$ within the current frame
        end if
        if $f < I$  then
            $A_f \leftarrow a_{i=f}$                      # Initialy, it tries all the actions
        else
            $A_f \leftarrow \underset{a_i}{\text{argmax}}[Q(a_i) + UCB(a_i)]$,      for $i = 0, 1, ..., I-1$, and $UCB(a_i)$ given by Eq. (2-7)
        end if
        $N(A_f) \leftarrow N(A_f) + 1$
        Schedule the packet transmission for the $Tx\_time$ instant
        Wake up  at the $Tx\_time$ instant and send the packet using the SF and transmit power corresponding to $A_f$
        $Tx\_prev\_frame \leftarrow True$
    else
        Sleep until the beginning of the next frame
    end if
end for
```

information will be ignored. If there was a packet transmission, the *RL-LoRa-UCB* algorithm will use the received reward (R_{f-1}) to update the value of the action selected during the previous frame (A_{f-1}). To this end, it employs the *sample-average* method introduced in Section 2.3.2.1 (Eq. (2-3)), which is represented in the line 4 of Algorithm 3.

After deciding if it will parse or not the information carried by the received beacon, and updating the value of the action selected during the previous frame (if necessary), the end device will check if there is a packet to transmit during the current frame. If there is data waiting to be transmitted, it will schedule a specific time for transmission (Tx_time) and will select the transmission parameters (the action A_f) for its data transmission. Otherwise, the end device will sleep until the beginning of the next frame. The value of Tx_time is set once during the whole node's operation time, it is selected at random during the first packet transmission (lines 7-9) and will be used in the subsequent ones.

Finally, the *RL-LoRa-UCB* algorithm uses the UCB's rule for selecting among the available actions, which is represented in the lines 10-14 of Algorithm 3. As can be observed, initially, all the available actions will be tried (lines 10-12), and after that, the action selection will be given by Eq. (2-8). After the action selection, the end device will schedule the packet transmission for the Tx_time instant within the current frame. It will sleep until this time when it will wake up to transmit their packet using the transmission parameters represented by the previously selected action (A_f).

After describing the proposed *RL-LoRa-UCB* algorithm, it will be evaluated under different parameter settings in Section 4.2.4.3. The next section will describe the simulation setup used through the rest of this chapter for evaluating all the proposed RL algorithms proposed.

4.2.4.2 Simulation setup for evaluating RL-LoRa's performance

The simulation setup used in the remainder of this chapter for evaluating the RL-LoRa protocol when using the proposed RL algorithms, employs the same topology and simulation parameters of Scenario 1 described in Section 3.2.2 (except for the ADR_ACK_LIMIT and ADR_ACK_DELAY parameters of the ADR mechanism of LoRaWAN, which are not used by RL-LoRa). Furthermore, two different cases are considered here:

- *Case 1:* The RL algorithm (i.e., the *RL-LoRa-UCB* algorithm) will control only the SFs. Thus, in this case, the set of actions corresponds to the actions a_0 - a_5 of the Table 4.3, each representing the available SFs in LoRaWAN. That is, for example, in Algorithm 3, the set of actions $\mathcal{A}$ would be given by $\mathcal{A} = \{a_0 = SF7, a_1 = SF8, ..., a_5 = SF12\}$. On the other hand, for this case, all packets will be transmitted with the maximal transmit power allowed in LoRaWAN (14dBm).

– *Case 2:* The RL algorithm will control both the SFs and the transmit power. Thus, in this case, the set of actions corresponds to the actions a_0 - a_{11} of the Table 4.3, each representing a combination of the available SFs in LoRaWAN and two of the allowed transmit powers (14 and 11 dBm). That is, for example, in Algorithm 3, the set of actions $\mathcal{A}$ would be given by $\mathcal{A} = \{a_0 = SF7/14dBm, a_1 = SF7/11dBm, ..., a_{10} = SF12/14dBm, a_{11} = SF12/11dBm\}$.

On the other hand, some extra parameters should be set during the evaluation of RL-LoRa's performance when using the proposed RL algorithms. One parameter is the frame duration (T_{frame}) of the RL-LoRa protocol, and the others are specific parameters of the RL algorithm in use (i.e., the exploration rate (c) of the *RL-LoRa-UCB* algorithm). All the evaluations presented in the remainder of this chapter, use a T_{frame} equal to 120 seconds. However, as the parameters of the RL algorithms vary according to the algorithm in use, they will be specified during the evaluation of each particular algorithm.

4.2.4.3 Tuning the parameters of the RL-LoRa-UCB algorithm

This section evaluates the performance of the proposed RL-LoRa protocol when using the *RL-LoRa-UCB* algorithm to locally adjust the transmission parameters of the end devices (RL-LoRa/*RL-LoRa-UCB*)[7], and compares it with the performance of the legacy LoRaWAN. The evaluations of *RL-LoRa-UCB* will consider the cases 1 and 2 described in the previous section as well as the parameters summarized in Table 4.4. It will allow identifying which configurations of the *RL-LoRa-UCB*'s parameters provide the best network performance in terms of the PER.

Table 4.4: Simulation parameters for evaluating the *RL-LoRa-UCB* algorithm.

Parameter	***Case 1***	***Case 2***
Actions set ($\mathcal{A}$)	$\{a_0, a_1, ..., a_5\}$ from Table 4.3	$\{a_0, a_1, ..., a_{11}\}$ from Table 4.3
Exploration rate (c)	0.1 and 0.2	0.1 and 0.2

RL-LoRa-UCB under *Case 1* (controlling only the SFs)

[7] The remainder of this thesis uses (RL-LoRa/*RL_Algorithm_Name*) to refer to the network that uses RL-LoRa as MAC protocol together with the *RL_Algorithm_Name* RL algorithm to control the transmission parameters at the end devices.

Fig. 4.5 shows the average PER versus distance to the central gateway under the legacy LoRaWAN (Scenario 1) and the proposed RL-LoRa/*RL-LoRa-UCB* MAC protocol (*Case 1*).

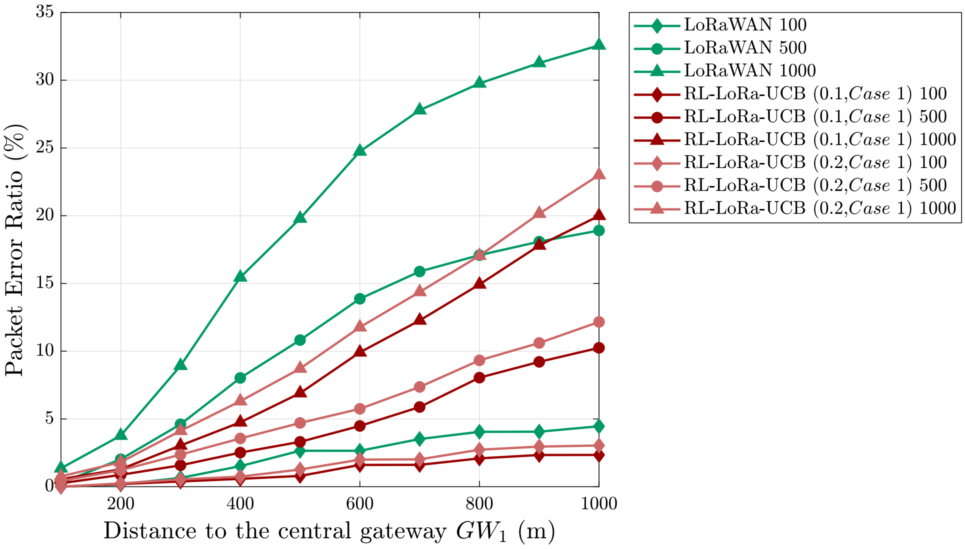

Figure 4.5: Average PER versus distance to the central gateway (GW_1) under LoRaWAN (Scenario 1) and RL-LoRa/*RL-LoRa-UCB* (*Case 1*). At the figure's legend, *"LoRaWAN N"* indicates that this curve corresponds to a LoRaWAN network with N nodes. Likewise, "*RL-LoRa-UCB* (c,*Case*) N" represents a network with N nodes that uses the RL-LoRa MAC protocol together with the *RL-LoRa-UCB* algorithm (using an exploration rate of c) and the configuration corresponding to the case "*Case*".

From the results in Fig. 4.5, it should be noticed that the PER under the proposed RL-LoRa/*RL-LoRa-UCB* is significantly lower than under LoRaWAN for all the distances from the central gateway. For example, when considering networks with 1000 nodes, RL-LoRa/*RL-LoRa-UCB* ($c = 0.1$) can reduce the PER from 33% to 20% at the cell edge, almost achieving the LoRaWAN's performance for a network with half of the nodes (LoRaWAN 500 nodes). Likewise, the results in Fig. 4.5, demonstrate that RL-LoRa/*RL-LoRa-UCB* ($c = 0.1$) enables the same PER (20%) as LoRaWAN, but at a distance 500 meters further from the central gateway. The reasons for the reliability and scalability improvements offered by RL-LoRa/*RL-LoRa-UCB* in comparison with LoRaWAN are explained below.

First, RL-LoRa/*RL-LoRa-UCB* can significantly reduce the packet collision by locally controlling the SF used by the end devices during each packet transmission. As explained in Section 4.2.4.1, initially, the end devices will try all the available actions. In this case (when controlling only the SFs), it means that the first six transmitted packets will use the six SFs available in LoRaWAN (i.e., the first packet is transmitted with SF7, the second packet with SF8, and so on). By doing this, after transmitting the first six packets, each end device will have a better understanding of which SFs provide the best communication channel with the central gateway. After this initial exploration, the *RL-LoRa-UCB* algorithm running at each node (the RL agent), will try to find the SF (action) that leads to the more reliable communication (allows receiving the best rewards). To this end, for each packet transmission, the *RL-LoRa-UCB* algorithm will try the action (SF) with the strongest potential to be optimal, that is, according to the line 13 of Algorithm 3. Once all the active nodes have "coordinated" their use of SFs (their actions), the network reliability will have been significantly improved, because the end devices will have learned the SF that avoid collisions between packets with the same SF. It increases network reliability by minimizing packet losses. As explained in Chapter 3, Section 3.1.1.3, when two packets with the same SF collide, at least one of these packets is going to be lost. In contrast, when colliding packets with different SFs, at least one of them will be correctly demodulated. More importantly, if there is a proper power control, both messages could be successfully received.

On the other hand, the good performance of RL-LoRa/*RL-LoRa-UCB* is also due to the fact that the proposed RL-LoRa MAC protocol helps the RL algorithms that are running at nodes, to rapidly coordinate their actions with each other. RL-LoRa states that the end devices will transmit always at the same instant within each frame (Tx_time). Thus, each node will have to coordinate its actions (SFs, in this case) always with the same competitors, which will take less time than if the competitors change at each frame.

From Fig. 4.5 one can conclude that, for the specific task of adjusting the SFs at the node's side through the proposed *RL-LoRa-UCB* algorithm, the best exploration rate is given by $c = 0.1$. It should be noticed that for a higher exploration rate ($c = 0.2$), *RL-LoRa-UCB* offers a worse PER, but still better than LoRaWAN's PER. This is because, in this case, allowing more exploration will enable the end devices to try, more frequently, SFs (actions) that lead to collisions between packets with the same SF. It will increase the probability of packet losses, affecting network reliability, as shown in Fig. 4.5. Finally, it should also be noticed that the PER decrease achieved by RL-LoRa/*RL-LoRa-UCB* over LoRaWAN is smaller with fewer nodes in the network (i.e., 100 nodes). This is because in smaller networks there are fewer packet collisions (i.e., the probability of having concurrent packet transmissions is lower) and, thus, the advantages of using the *RL-LoRa-UCB* algorithm for controlling the SFs are less perceptible.

Below, the *RL-LoRa-UCB* algorithm is evaluated under *Case 2*, when it has to learn the best action from a larger action set.

RL-LoRa-UCB under *Case 2* (controlling both SFs and transmit power)

Fig. 4.6 shows the average PER versus distance to the central gateway under the legacy LoRaWAN (Scenario 1) and the proposed RL-LoRa/*RL-LoRa-UCB* (*Case 2*). As depicted in this figure, in this case, RL-LoRa/*RL-LoRa-UCB* also offers a better network performance compared to LoRaWAN, outperforming this latter in terms of the average PER for all the distances from the gateway. On the other hand, the results in Fig. 4.6 show that, in this case, the best exploration rate is also given by $c = 0.1$. As in the *Case 1*, a higher exploration rate ($c = 0.2$) leads to a worse PER for all the distances from GW_1.

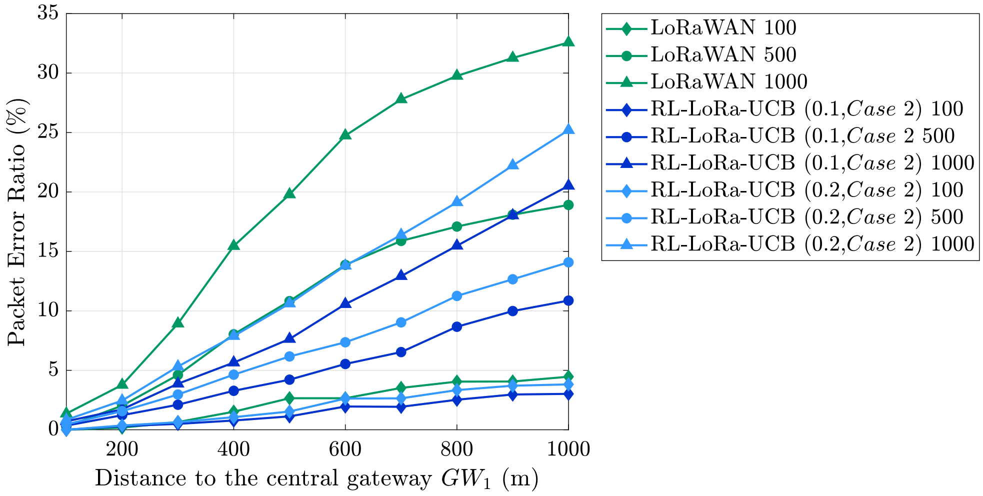

Figure 4.6: Average PER versus distance to the central gateway (GW_1) under LoRaWAN (Scenario 1) and RL-LoRa/*RL-LoRa-UCB* (*Case 2*). At the figure's legend, *"LoRaWAN N"* indicates that this curve corresponds to a LoRaWAN network with N nodes. Likewise, "*RL-LoRa-UCB* (c,*Case*) N" represents a network with N nodes that uses the RL-LoRa MAC protocol together with the *RL-LoRa-UCB* algorithm (using an exploration rate of c) and the configuration corresponding to the case "*Case*".

The reasons for the improvements that *RL-LoRa-UCB* offers over LoRaWAN were already explained in the previous subsection (RL−LoRa−UCB under *Case 1* (controlling only the SFs)). The difference here (*Case 2*) is that *RL-LoRa-UCB* has to lead with a larger set of actions, including not only SFs but also two possibilities of transmit power (14 and 11 dBm). According to the rules for discarding packets when a collision occurs (explained in Chapter 3, Section 3.1.1.3), controlling the transmit power, in addition to the SFs, would offer better network reliability. This is because the end devices will learn not only the SF that avoids collisions between packets with the same SF but also the transmit power that ensures proper demodulation of the colliding packets in case a collision has occurred. That is, controlling the transmit power will help to ensure the required SINR for correctly demodulating the colliding messages, which depends on the SF used by the packets involved in the collision [4].

The next section will compare the performance of the proposed RL-LoRa/*RL-LoRa-UCB* MAC protocol under cases 1 and 2. It will allow investigating if RL-LoRa/*RL-LoRa-UCB* can take advantage of the joint control of SFs and transmit power at the end devices.

Comparing the performance of RL-LoRa-UCB under *Cases 1 and 2*

Fig. 4.7 shows the performance comparison (in terms of the average PER) between the legacy LoRaWAN (Scenario 1) and the proposed RL-LoRa/*RL-LoRa-UCB* MAC protocol (*Cases 1 and 2*), when this latter is using the best parameters setting, that is, $c = 0.1$ for both cases. As can be observed, contrary to the expected result, the average PER under the *Case 2* is worse than that under the *Case 1* for all the distances from the central gateway. The reasons for these results rely on the approach that *RL-LoRa-UCB* uses for updating the action values (the *sample-average* method), which constitutes the weakness of this algorithm.

As shown in Algorithm 3 (line 4), the *sample-average* method uses a step-size that decreases with the number of times that each action has been selected ($\frac{1}{N(A_{f-1})}$). However, in this case, where the received rewards depend on the behavior of all the nodes in the network, it could be better to use an update rule that gives more importance to recent rewards, than to long-past rewards [2]. On the other hand, although the law of large numbers ensures the convergence of the action values estimate ($Q(a_i)$ for $a_i \in \mathcal{A}$) to their real values ($q_*(a_i)$) when using the *average-sample* method [2], the convergence time will depend on the number of considered actions. The higher set of actions, the more time the *RL-LoRa-UCB* algorithm needs for learning the real value of its actions and then for identifying the best one. For that reason, the proposed RL-LoRa/*RL-LoRa-UCB* offers a worse PER when adjusting both SFs and transmit power (*Case 2*) than when adjusting only the SFs (*Cse 1*): for both cases, the *RL-LoRa-UCB* algorithm is executed 720 times (there are 720 time-steps)[8], but under *Case 2*, the set of actions is twice that under *Case 1*.

[8] In this case, there are 720 time-steps because the simulation time is 24 hours and the *RL-LoRa-UCB* is executed every 120 seconds (each time a new beacon is received).

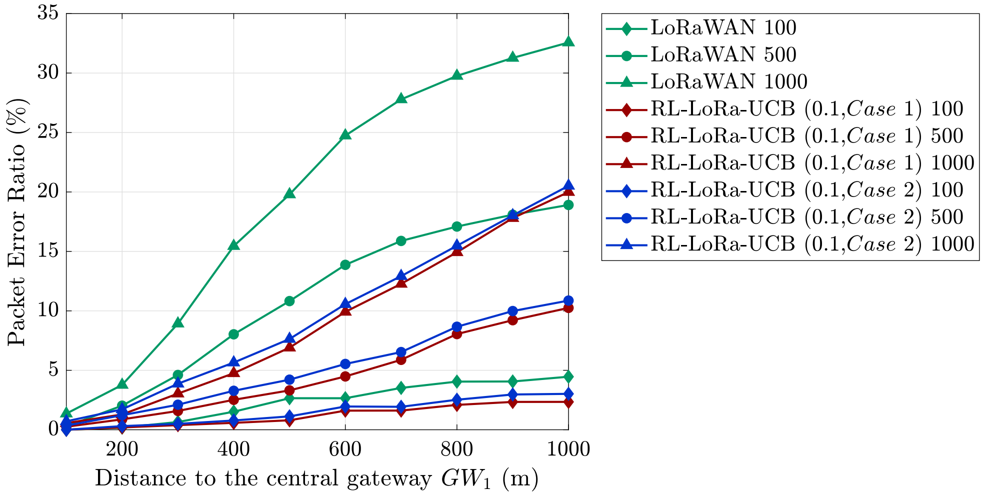

Figure 4.7: Average PER versus distance to the central gateway (GW_1) under LoRaWAN (Scenario 1) and RL-LoRa/*RL-LoRa-UCB* (*Cases 1 and 2*). At the figure's legend, *"LoRaWAN N"* indicates that this curve corresponds to a LoRaWAN network with N nodes. Likewise, "*RL-LoRa-UCB* (c,$Case$) N" represents a network with N nodes that uses the RL-LoRa MAC protocol together with the *RL-LoRa-UCB* algorithm (using an exploration rate of c) and the configuration corresponding to the case "$Case$".

To conclude, this section has demonstrated the value of the proposed RL-LoRa protocol for controlling the MAC layer in LoRa based networks. It has shown that using RL approaches for locally adjusting the transmission parameters at the end devices is a prominent solution if the reward information is efficiently carried from gateways to the RL algorithms that are being executed at nodes. The *RL-LoRa-UCB* algorithm has shown that RL approaches perform well in the task of deciding the best set of transmission parameters at each active node in a LoRaWAN network. The simulation results in Fig. 4.7 demonstrate that RL-LoRa/*RL-LoRa-UCB* overcomes the performance of the legacy LoRaWAN under both *Case 1 and 2*, and for all the distances from the central gateway.

The remainder of this chapter tries to improve the results obtained for the proposed RL-LoRa/*RL-LoRa-UCB* MAC protocol. To this end, the next sections will introduce two RL algorithms that employ a more appropriate rule for updating the action values under nonstationary environments, which will allow taking advantage of the joint control of SFs and transmit power at the end devices.

4.2.4.4 The RL-LoRa-QL algorithm

The second RL algorithm proposed in this thesis for locally adjusting the transmission parameters of the end devices, is based on the previously introduced *RL-LoRa-UCB* algorithm, but defines different rules for both selecting among the available actions and for updating their values. Specifically, it substitutes the *RL-LoRa-UCB*'s rule for action selection (Algorithm 3, lines 10-14) by the ϵ-*greedy* approach described in Chapter 2 (Section 2.3.2.2). Likewise, it employs the action values update rule proposed by the stateless Q-Learning algorithm (Chapter 2, Section 2.3.4, Eq. 2-22), instead of the *sample-average* method used by *RL-LoRa-UCB* (Algorithm 3, line 4). This new algorithm was named *RL-LoRa-QL*, and is represented in Algorithm 4.

As *RL-LoRa-QL* is based on the *RL-LoRa-UCB* algorithm, it follows the same operation of the latter, which was already explained in Section 4.2.4.4. Likewise, by comparing Algorithm 3 with Algorithm 4, one can notice that *RL-LoRa-QL* makes the same parameter's initialization that *RL-LoRa-UCB*, and also uses most of the latter's parameters, except the exploration rate (c), which is substituted by the following parameters:

- The learning rate $\alpha \in [0, 1)$: It is used when updating the action values (Algorithm 4, line 4) and corresponds to the *StepSize* parameter introduced in Chapter 2, Section 2.3.2.1, Eq. (2-4). Since α is multiplying the error in the action value estimate ($R_{f-1} - Q(A_{f-1})$), it is valuing the information (knowledge) gained during each action value update. In that sense, setting α to zero means that the action values will never be updated, thereby nothing is learned. In contrast, setting α to a high value will accelerate the learning process by attributing a high weight to the knowledge gained at each action value update. However, it should be noticed that the value of α can also affect the convergence of the RL algorithm. Although a high α will lead to faster learning, it may also find harder to discover the real value of the considered actions; that is, the *RL-LoRa-QL* algorithm will take longer to converge.
- The epsilon parameter is used by the action selection rule of the *RL-LoRa-QL* algorithm to control the degree of the exploration.

From Algorithm 4, it should be noticed that the action selection rule used by *RL-LoRa-QL* is very simple (Algorithm 4, lines 11-15). It corresponds to the ϵ-*greedy* method, establishing that the highest value action will be selected with probability $1-\epsilon$, while a random action, with a probability ϵ. To this end, each time the *RL-LoRa-QL* algorithm will select an action, it first generates a

Algorithm 4: The *RL-LoRa-QL* algorithm

```
Algorithm parameters:
    Action set: $\mathcal{A} = \{a_0, a_1, ..., a_{I-1}\}$, where $I = |\mathcal{A}|$ is the total number of available actions;
    $T_{frame}$: frame duration (time inter beacon)
    $Q(a_i)$: actions value estimate, for $i = 0, 1, ..., I - 1$;
    $A_f$: action selected during the frame $f$;
    $R_f$: reward corresponding to $A_f$;
    $Tx\_prev\_frame$: indicates if there was a packet transmission during the previous frame
    $Pkt\_to\_Tx$: indicates  if there is a packet to transmit on the current frame
    $Tx\_time$: specifies the transmission instant for this node
    $\alpha \in [0, 1)$: learning rate;
    $\epsilon \in [0, 1)$: controls the degree of exploration
    $Rnd\_Nb \in [0, 1)$: random number generated during action selection
Initialize:
    $Q(a_i) = 0$, for $i = 0, 1, ..., I - 1$;
    $Tx\_prev\_frame$ = False;

 1: for each frame $f$ do
 2:     Receive the beacon message
 3:     if $Tx\_prev\_frame = True$ then
 4:         $Q(A_{f-1}) \leftarrow Q(A_{f-1}) + \alpha[R_{f-1} - Q(A_{f-1})]$        # By Eq. (2-22)
 5:     end if
 6:     if $Pkt\_to\_Tx = True$ then
 7:         if $f = 0$ then
 8:             $Tx\_time \leftarrow$ rand(0,$T_{frame}$)   # Select, at random, a value for $Tx\_time$ within the current frame
 9:         end if
10:         $Rnd\_Nb \leftarrow$ rand[0,1)             # Generate a random number in the interval [0,1)
11:         if $Rnd\_Nb < \epsilon$ then
12:             $A_f \leftarrow a_{i=rand[0,I-1]}$                        # Select a random action
13:         else
14:             $A_f \leftarrow \underset{a_i}{\text{argmax}}\, Q(a_i)$, for $i = 0, 1, ..., I - 1$  # Select the action with the highest value
15:         end if
16:         Schedule the packet transmission for the $Tx\_time$ instant
17:         Wake up  at the $Tx\_time$ instant and send the packet using the SF and transmit power corresponding to $A_f$
18:         $Tx\_prev\_frame \leftarrow True$
19:     else
20:         Sleep until the beginning of the next frame
21:     end if
22: end for
```

random number $Rnd\_Nb \in [0, 1)$ (Algorithm 4, line 11). Then, if Rnd_Nb is lower than ϵ, an action is selected at random from the set of available actions

(Algorithm 4, line 13). Otherwise, the action with the highest estimated value will be chosen (Algorithm 4, line 14).

The next section will evaluate the performance of the proposed *RL-LoRa-QL* algorithm under the cases described in Section 4.2.4.2 (*Cases 1and 2*) and when varying the parameters α and ϵ. It will allow identifying the best configuration of parameters for *RL-LoRa-QL* in the task of locally adjusting the transmission parameters of the end devices. Furthermore, the next section will compare the network performance under the proposed *RL-LoRa-QL* and *RL-LoRa-UCB* algorithms. It will investigate if using a more appropriate rule for updating the action values, really enables taking advantage of the joint control of SFs and transmit power.

4.2.4.5 Tuning the parameters of the RL-LoRa-QL algorithm

This section evaluates the performance of the proposed RL-LoRa MAC protocol when using *RL-LoRa-QL* to locally control the transmission parameters of the end devices (RL-LoRa/*RL-LoRa-QL*), and compares it with RL-LoRa/*RL-LoRa-UCB*, for the latter's best configuration of parameters ($c = 0.1$). The evaluation presented here consider the cases 1 and 2 described in Section 4.2.4.2 as well as the parameters summarized in Table 4.5.

Table 4.5: Simulation parameters for evaluating the *RL-LoRa-QL* algorithm.

Parameter	***Case 1***	***Case 2***
Actions set ($\mathcal{A}$)	$\{a_0, a_1, ..., a_5\}$ from Table 4.3	$\{a_0, a_1, ..., a_{11}\}$ from Table 4.3
Learning rate (α)	0.1 and 0.2	0.1 and 0.2
Epsilon (ϵ)	0.1 and 0.3	0.1 and 0.3

RL-LoRa-QL under *Case 1* (controlling only the SFs)

Fig. 4.8 shows the average PER versus distance to the central gateway under the proposed RL-LoRa/*RL-LoRa-QL* when considering *Case 1*.

As depicted in Fig. 4.8, *RL-LoRa-QL* provides the best performance (the lower average PER) for $\alpha = 0.2$ and $\epsilon = 0.1$, which constitutes the best configuration of the *RL-LoRa-QL*'s parameters in this particular case. It should be noticed that allowing a higher degree of exploration ($\epsilon = 0.3$) leads to a network performance significantly worse. This is because end devices will try random actions with a higher probability, increasing the chance of using

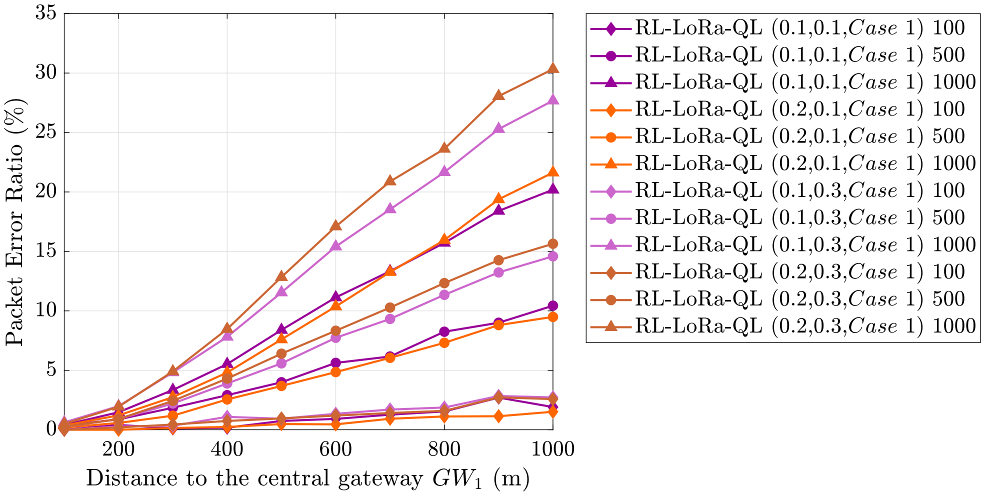

Figure 4.8: Average PER versus distance to the central gateway (GW_1) under RL-LoRa/*RL-LoRa-QL* (*Case 1*). At the figure's legend, "*RL-LoRa-QL* (α,ϵ,*Case*) N" indicates that this curve corresponds to a network with N nodes that uses the RL-LoRa MAC protocol together with the *RL-LoRa-QL* algorithm (using a learning rate of α and an epsilon of ϵ) and the configuration corresponding to the case "*Case*".

actions (SFs) that lead to bad rewards (collisions between packets using the same SF). Thus, leading to higher PERs, as shown in Fig. 4.8.

On the other hand, when ϵ is fixed (i.e., $\epsilon = 0.1$), the variations on the *RL-LoRa-QL*'s performance are given by the effect of the learning rate α. As can be observed in Fig. 4.8, in this case (RL-LoRa/*RL-LoRa-QL*, *Case 1*, $\epsilon = 0.1$), a faster learning rate ($\alpha = 0.2$) provides the best network performance, especially, for small networks (100 nodes). This is because, in small networks, there are fewer agents (nodes) to coordinate their actions with each other. Thus, a faster learning rate will accelerate the learning process for each RL agent (they will find their best actions earlier) without affecting its convergence. That is, the end devices will know the SFs (actions) that provide the best communication channel with the central gateway earlier and will exploit these actions to ensure reliable communication.

RL-LoRa-QL under *Case 2* (controlling both SFs and transmit power)

Fig. 4.9 shows the average PER versus distance to the central gateway under RL-LoRa/*RL-LoRa-QL* (*Case 2*).

By comparing Fig. 4.9 with Fig. 4.8, one can conclude that *RL-LoRa-QL*, in terms of the parameters used, follows a similar behavior under *Cases 1 and 2*. For example, for both cases, the best PER is achieved with $\alpha = 0.2$ and $\epsilon = 0.1$ for all the distances from the central gateway. The explanation

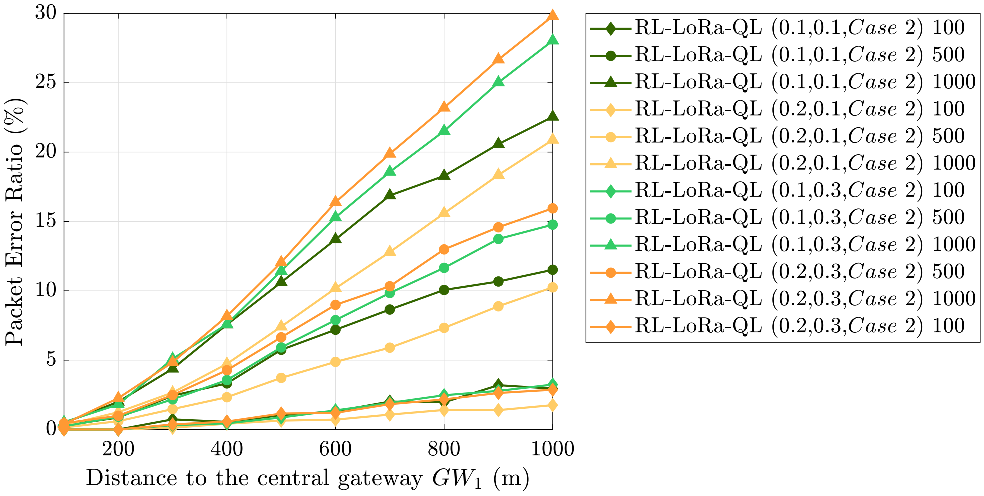

Figure 4.9: Average PER versus distance to the central gateway (GW_1) under RL-LoRa/*RL-LoRa-QL* (*Case 2*). At the figure's legend, "*RL-LoRa-QL* (α,ϵ,*Case*) N" indicates that this curve corresponds to a network with N nodes that uses the RL-LoRa MAC protocol together with the *RL-LoRa-QL* algorithm (using a learning rate of α and an epsilon of ϵ) and the configuration corresponding to the case "*Case*".

for these results were already presented in the previous subsection.

RL-LoRa-UCB and RL-LoRa-QL comparison under *Cases 1 and 2*

Fig. 4.10 compares the network performance of RL-LoRa/*RL-LoRa-UCB* and RL-LoRa/*RL-LoRa-QL* in terms of the average PER under the *Cases 1 and 2* described in Section 4.2.4.2. From this figure, it should be noticed that for small networks (100 nodes), *RL-LoRa-QL* outperforms *RL-LoRa-UCB* in terms of the average PER, for all the distances from the central gateway. Although the improvements offered by *RL-LoRa-QL* over *RL-LoRa-UCB* are not that significant (a 2% lower PER at the cell edge for a network with 100 nodes under *Case 2*), the results in Fig. 4.10 demonstrate that is possible to improve the network performance by using an RL algorithm with a more appropriate rule for updating the action values. Specifically, the action values update rule used by *RL-LoRa-QL* uses a fix *StepSize* parameter (α). It allows assigning more weight to recent rewards than to long-past rewards, which is more appropriate when the RL agents have to interact with a nonstationary environment [2], as in this case.[9]

[9] In this case, the environment is considered nonstationary because the rewards received by the RL agent running at an end device, are not *i.i.d*: they depend on the behavior of that node, but also on the behavior of other nodes. Thus, the environment can not be considered stationary anymore.

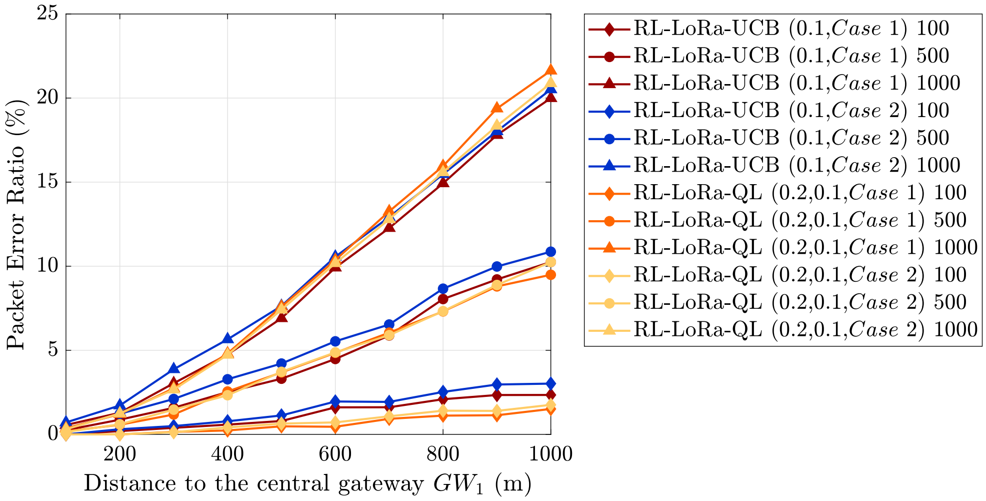

Figure 4.10: Average PER versus distance to the central gateway (GW_1) under RL-LoRa/*RL-LoRa-UCB* and RL-LoRa/*RL-LoRa-QL* (*Cases 1 and 2*). At the figure's legend, "*RL-LoRa-UCB*/*RL-LoRa-QL* (α,ϵ,*Case*) N" indicates that this curve corresponds to a network with N nodes that uses the RL-LoRa MAC protocol together with the *RL-LoRa-UCB*/*RL-LoRa-QL* algorithm (using a learning rate of α and an epsilon of ϵ) and the configuration corresponding to the case "*Case*".

On the other hand, Fig. 4.10 shows that when considering larger networks (500 and 1000 nodes), the PER under *RL-LoRa-QL* becomes worse than that under *RL-LoRa-UCB* for almost all the distances from the central gateway. There are two main reasons for these results. The first reason is that, although *RL-LoRa-QL* uses a more appropriate method for updating the action values, it still has a weakness compared to *RL-LoRa-UCB*: the rule it uses for selecting among the available action (the ϵ-*greedy* approach). Although ϵ-*greedy* gives the RL agent the opportunity to eventually try all the actions (a random action will be selected with probability ϵ), this random exploration might lead the agent to try actions that will not give a good reward, even when the agent has already identified its best action. In that sense, the action selection rule used by *RL-LoRa-UCB* is more efficient than the ϵ-*greedy* method, because it favors the exploration of actions with a strong potential to give a good reward. The second reason is that, for the specific task of locally adjusting the transmission parameters of LoRaWAN end devices, the effect of employing a weak approach for controlling the exploration will become more important as the number of end devices in the network increases. This is because in large networks there are more packet collisions, which will lead to packet losses if the end devices do not use the appropriate SF and transmit power. That is, if they do not exploit their knowledge by using the highest value actions, or actions with a strong potential to have an optimal value. In this way, letting the end devices to select actions at random, as the ϵ-*greedy* method does, will increase the probability of losing the packets involved in a collision. It is even more critical for large networks where a large number of packet collisions occur.

Finally, according to the results obtained when evaluating the performance of RL-LoRa/*RL-LoRa-UCB* and RL-LoRa/*RL-LoRa-QL*, a potential solution to achieve a better network performance is to combine their strengths in a new RL algorithm. That is, to combine in a single algorithm the *RL-LoRa-UCB*'s method for selecting among the available actions (the UCB's action selection algorithm) with the approach of *RL-LoRa-QL* for updating the action values (the action values update rule of stateless Q-Learning). It is done by the last RL algorithm proposed in this thesis, which is introduced in the next section.

4.2.4.6 The RL-LoRa-QL-UCB algorithm

This section presents the *RL-LoRa-QL-UCB* algorithm, which combines the strengths of *RL-LoRa-UCB* and *RL-LoRa-QL* to improve their performance in the task of locally adjusting the transmission parameters of LoRaWAN end devices. The new resulting algorithm is represented in Algorithm 5. As can be observed, it follows the same operation that the *RL-LoRa-UCB* and *RL-LoRa-QL* algorithms, and also uses their parameters, which were already described in the sections 4.2.4.1 and 4.2.4.4, respectively.

In sections 4.2.4.3 and 4.2.4.5, were already identified the best parameter settings of the *RL-LoRa-UCB*'s rule for selecting actions (the exploration rate $c = 0.1$) and of the *RL-LoRa-QL*'s approach for updating the action values (learning rate $\alpha = 0.2$). Thus, in this section, it will not be necessary tuning the parameters of the *RL-LoRa-QL-UCB* algorithm. In the next section, it will be evaluated under the *Cases 1 and 2* described in Section 4.2.4.2 and using the best parameters configuration of the *RL-LoRa-UCB* and *RL-LoRa-QL* algorithms ($c = 0.1$ and $\alpha = 0.2$).

Performance comparison among RL-LoRa-UCB, RL-LoRa-QL, and RL-LoRa-QL-UCB under *Cases 1 and 2*

Fig. 4.11 shows performance of RL-LoRa/*RL-LoRa-UCB*, RL-LoRa/*RL-LoRa-QL*, and RL-LoRa/*RL-LoRa-QL-UCB* under *Cases 1 and 2* described in Section 4.2.4.2. The specific simulation parameters used to obtain these evaluation results are summarized in Table 4.6. As can be observed, they correspond to the best setting of parameters of the proposed RL algorithms, which were already identified in the previous sections.

Algorithm 5: The *RL-LoRa-QL-UCB* algorithm

```
Algorithm parameters:
    Action set: $\mathcal{A} = \{a_0, a_1, ..., a_{I-1}\}$, where $I = |\mathcal{A}|$ is the total number of
    available actions;
    $T_{frame}$: frame duration (time inter beacon)
    $Q(a_i)$: actions value estimate, for $i = 0, 1, ..., I - 1$;
    $N(a_i)$: actions usage counter, for $i = 0, 1, ..., I - 1$;
    $A_f$: action selected during the frame $f$;
    $R_f$: reward corresponding to $A_f$;
    $Tx\_prev\_frame$: indicates if there was a packet transmission during
    the previous frame
    $Pkt\_to\_Tx$: indicates  if there is a packet to transmit on the current
    frame
    $Tx\_time$: specifies the transmission instant for this node
    $\alpha \in [0, 1)$: learning rate;
    $c > 0$: exploration rate;
Initialize:
    $Q(a_i) = N(a_i) = 0$, for $i = 0, 1, ..., I - 1$;
    $Tx\_prev\_frame$ = False;

for each frame $f$ do
    Receive the beacon message
    if $Tx\_prev\_frame = True$ then
        $Q(A_{f-1}) \leftarrow Q(A_{f-1}) + \alpha[R_{f-1} - Q(A_{f-1})]$        # By Eq. (2-22)
    end if
    if $Pkt\_to\_Tx = True$ then
        if $f = 0$ then
            $Tx\_time \leftarrow$ rand(0,$T_{frame}$)   # Select, at random, a value for
            $Tx\_time$ within the current frame
        end if
        if $f < I$  then
            $A_f \leftarrow a_{i=f}$                    # Initialy, it tries all the actions
        else
            $A_f \leftarrow \underset{a_i}{\text{argmax}}[Q(a_i) + UCB(a_i)]$,      for $i = 0, 1, ..., I - 1$, and
            $UCB(a_i)$ given by Eq. (2-7)
        end if
        $N(A_f) \leftarrow N(A_f) + 1$
        Schedule the packet transmission for the $Tx\_time$ instant
        Wake up  at the $Tx\_time$ instant and send the packet using the
        SF and transmit power corresponding to $A_f$
        $Tx\_prev\_frame \leftarrow True$
    else
        Sleep until the beginning of the next frame
    end if
end for
```

As depicted in Fig. 4.11, *RL-LoRa-QL-UCB* overcomes the performance of the previously proposed RL algorithms (*RL-LoRa-UCB* and *RL-LoRa-QL*) in terms of the average PER for all the distances from the central gateway,

Table 4.6: Simulation parameters for the comparison among the *RL-LoRa-UCB*, *RL-LoRa-QL*, and *RL-LoRa-QL-UCB* algorithms.

Parameter	***Case 1***	***Case 2***
Actions set ($\mathcal{A}$)	$\{a_0, a_1, ..., a_5\}$ from Table 4.3	$\{a_0, a_1, ..., a_{11}\}$ from Table 4.3
Exploration rate (c)	0.1	0.1
Learning rate (α)	0.2	0.2
Epsilon (ϵ)	0.1	0.1

and under both *Case 1* and *Case 2*. Specifically, for a network with 500 nodes, RL-LoRa/*RL-LoRa-QL-UCB* (c = 0.1 and α = 0.2) can reduce the PER under RL-LoRa/*RL-LoRa-QL* (α = 0.2 and ϵ = 0.1) in almost a 4% at the cell edge.

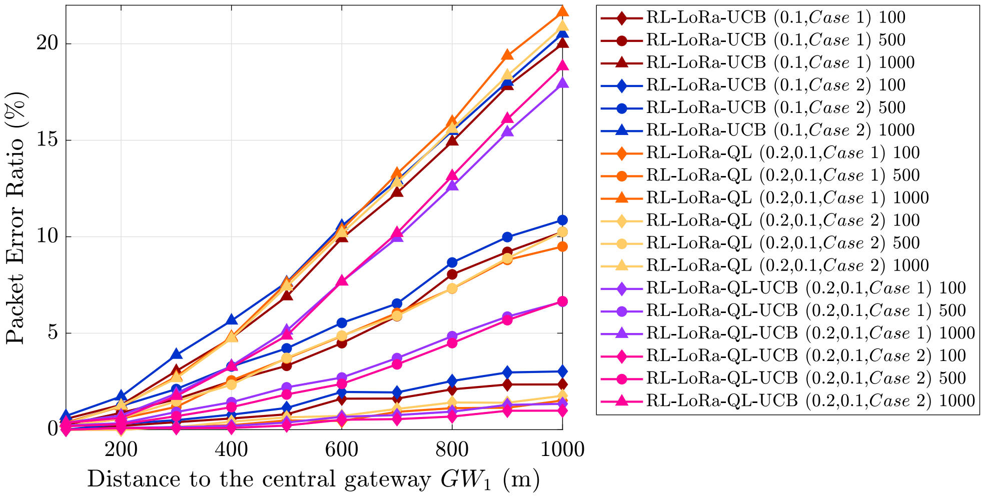

Figure 4.11: Average PER versus distance to the central gateway (GW_1) under RL-LoRa/*RL-LoRa-UCB*, RL-LoRa/*RL-LoRa-QL*, and RL-LoRa/*RL-LoRa-QL − UCB* (*Cases 1 and 2*). At the figure's legend, "*RL-LoRa-UCB*/*RL-LoRa-QL*/*RL-LoRa-QL-UCB* (α,ϵ,*Case*) N" indicates that this curve corresponds to a network with N nodes that uses the RL-LoRa MAC protocol together with the *RL-LoRa-UCB*/*RL-LoRa-QL*/*RL-LoRa-QL-UCB* algorithm (using a learning rate of α and an epsilon of ϵ) and the configuration corresponding to the case "*Case*".

The results in Fig. 4.11 demonstrate the effectiveness of combining the strengths of *RL-LoRa-UCB* and *RL-LoRa-QL* in a single RL algorithm. *RL-LoRa-QL-UCB* has demonstrated that using the appropriate approaches for updating the action values and for selecting among the available actions (properly controlling the trade-off between exploration and exploitation), is essential to ensure the good performance of the RL algorithms in the task of locally controlling the transmission parameter of LoRaWAN end devices. The advantages of individually using the *RL-LoRa-QL-UCB*'s action selection method (the UCB's action selection rule) and its approach for updating the action values (the action update rule of stateless Q-Learning), were already explained in sections 4.2.4.3 and 4.2.4.5, respectively. On the other hand, the advantages of combining these approaches have been discussed in this section and can be observed in Fig. 4.11.

Finally, from Fig. 4.11 it should be noticed that among the proposed RL algorithms, only *RL-LoRa-QL-UCB* can take advantage of the joint control of SFs and transmit power. In networks with 100 and 500 nodes, it provides under *Case 2* a PER that is slightly better than that under *Case 1* for all the distances from the central gateway.

The next section will carry out a performance comparison among the MAC protocol introduced in this chapter (RL-LoRa), LoRaWAN, and the recently proposed RS-LoRa protocol [37]. It will show the advantages of RL-LoRa over the state-of-the-art MAC protocols that try to improve the performance of LoRaWAN.

4.2.5 Performance comparison among the RL-LoRa, LoRaWAN and RS-LoRa protocols under single gateway scenarios

This section carries out a performance comparison among the novel RL-LoRa MAC protocol (when using all the proposed RL algorithms: *RL-LoRa-UCB*, *RL-LoRa-QL*, and *RL-LoRa-QL-UCB*), and the legacy LoRaWAN and RS-LoRa protocols. To this end, various important metrics in the context of the IoT networks are used, including the average PER, average network delay, and network throughput, which are defined as in Chapter 3 (Section 3.2.3). Table 4.7 summarizes the simulation setup for the protocols involved in the performance comparison presented in what follows.

From Table 4.7, it should be noticed that the RL-LoRa protocol is evaluated only under *Case 2*. This is because *Case 2* is a general case allowing the adjustment of not only the SFs but also the transmit power at the end devices. Likewise, the evaluation of the LoRaWAN and RS-LoRa protocols is carried out under Scenario 1 described in Chapter 3 (Section 3.2.3). On the other hand, it is important highlight that the T_{frame} paramater of RS-LoRa, has the same meaning of the T_{frame} parameter of the proposed RL-LoRa protocol: setting the frame duration.

Finally, RS-LoRa was selected for this performance comparison due to its similarities with the proposed RL-LoRa protocol. For example, RS-LoRa also

Table 4.7: Simulation setup for comparing the RL-LoRa, LoRaWAN, and RS-LoRa protocols under single gateway scenarios.

MAC protocol	**Simulation septup**
RL-LoRa	- *Case 2* and Scenario 1 - $T_{frame} = 120$ seconds - Exploration rate: $c = 0.1$ - Learning rate: $\alpha = 0.2$ - Epsilon: $\epsilon = 0.1$
LoRaWAN	- Scenario 1
RS-LoRa	- Scenario 1 - $T_{frame} = 60$ seconds

uses a *synchronous* and *distributed* approach for adjusting the transmission parameters at the end devices. Besides, although RS-LoRa uses a different[10] approach for adapting the transmission parameters at the end devices, it is the earliest work trying to it dynamically to improve the performance of LoRaWAN. Thus, due to its similarities with RL-LoRa, to date, RS-LoRa is the best candidate for assessing the value of the proposed RL-LoRa MAC protocol through a performance comparison.

Average Packet Error Ratio

Fig. 4.12 shows the average PER under LoRaWAN, RS-LoRa, and the proposed RL-LoRa (when using the *RL-LoRa-UCB*, *RL-LoRa-QL*, and *RL-LoRa-QL-UCB* algorithms) MAC protocols. As can be observed, RL-LoRa provides a lower average PER than the legacy LoRaWAN and RS-LoRa MAC protocols, under all the proposed RL algorithms in this thesis for adjusting the transmission parameters at the end devices. Specifically, for a network with 1000 nodes, the best RL algorithm proposed by this thesis ($RL\text{-}LoRa\text{-}QL - UCB$, $\alpha = 0.2$ and $c = 0.1$), offers an average PER that is a 15% lower than the average PER provided by LoRaWAN.

From Fig. 4.12 it should also be noticed that RL-LoRa, in addition to significantly improving the performance of LoRaWAN in terms of the average PER, it also improves the average PER under RS-LoRa for all the considered network sizes (100, 500 and 1000 nodes). These evaluation results demonstrate the value of RL-LoRa to control the MAC layer of LoRaWAN end devices, and the appropriateness of using RL algorithms to control the latter's transmission parameters.

[10] RS-LoRa nodes receive through beacons a specific scheduling of the transmission parameters they should use for the packet transmissions within each frame (it is scheduled by the gateway), while RL-LoRa nodes can learn by themselves the best set of transmission parameters through RL techniques.

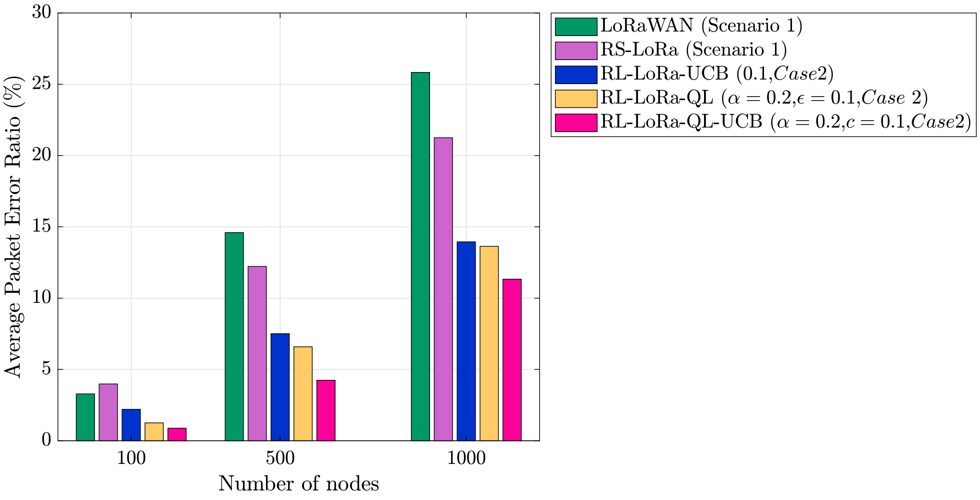

Figure 4.12: Average PER under LoRaWAN, RS-LoRa, RL-LoRa.

Network throughput

Fig. 4.13 presents the network throughout under the studied protocols. As can be seen, the best network throughput is achieved under the proposed RL-LoRa MAC protocol, specifically when using the *RL-LoRa-QL-UCB* algorithm to control the transmission parameters at the end devices. This figure also demonstrates how RL-LoRa overcomes the LoRaWAN and RS-LoRa protocols in terms of the achievable network throughput.

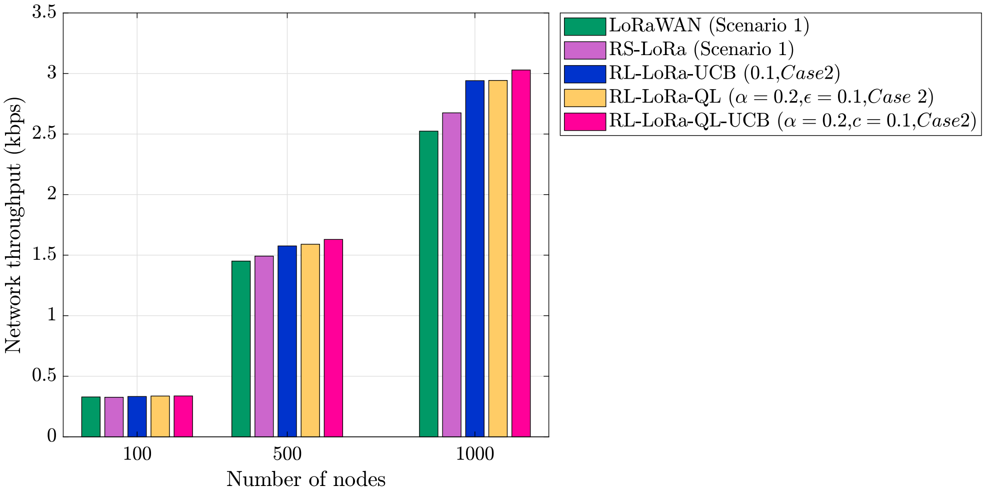

Figure 4.13: Network throughput under LoRaWAN, RS-LoRa, RL-LoRa.

Average network delay

Fig. 4.14 shows the average network delay under LoRaWAN, RS-LoRa, RL-LoRa. It demonstrates that both RS-LoRa and the proposed RL-LoRa

MAC protocols, present a notable increase in the average network delay when comparing with LoRaWAN. The reason for these results is the way on which the average network delay is calculated for each node (explained in Chapter 3, Section 3.2.3). As in RS-LoRa and RL-LoRa based networks, nodes transmit unconfirmed traffic (they do not request ACKs), the average network delay, in this case, corresponds to the time elapsed from a packet transmission within a frame, until the reception of the beacon staring in the next frame. For that reason, as depicted in Fig. 4.14, the average network delay under RS-LoRa and RL-LoRa is around the value set for the T_{frame} parameter in both cases, 60 and 120 seconds respectively.

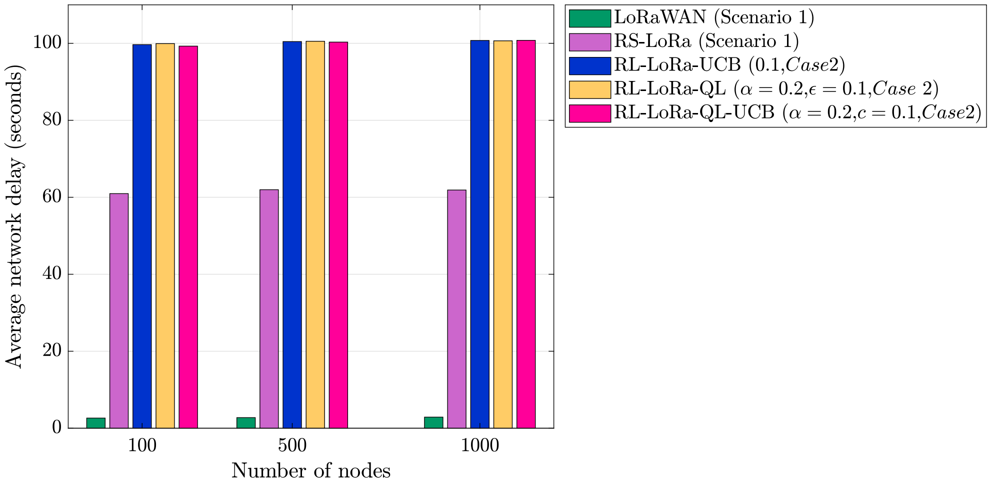

Figure 4.14: Average network delay under LoRaWAN, RS-LoRa, RL-LoRa.

By comparing the figures 4.14, 4.13, and 4.12, one can notice the trade-off between average network delay and network reliability. That is, although RL-LoRa significantly improves the network performance in terms of the average PER and the achievable network throughput, it also increases the network delay due to its synchronous nature. In RL-LoRa, nodes can transmit their messages only after receiving a beacon, and at a specific instant within the frame (Tx_time). Although it is possible to decrease such a network delay by using a lower T_{frame}, it will also lead to higher energy consumption because nodes will have to wake up to receive beacon messages more frequently, even when they do not have a new packet to transmit.

Energy Efficiency

Finally, the LoRaWAN, RS-LoRa, and RL-LoRa protocols are compared in terms of how efficiently their nodes use the energy they are provided through batteries. To this end, a new metric is introduced here. It is the *Energy*

Efficiency which is defined as the overall PDR over the average energy cost of transmitting a packet (the average energy consumption defined in Chapter 3, Section 3.2.3).

Fig. 4.15, shows the energy efficiency under the studied protocols. From this figure, it should be noticed that both RS-LoRa and RL-LoRa are less energy efficient than LoRaWAN under all the considered number of nodes. This is because these protocols introduce additional energy consumption by scheduling nodes to use higher SFs and to receive beacons. On the other hand, it should be noticed that the energy efficiency of LoRaWAN significantly decreases with the number of nodes in the network. This is because, in larger networks, there are more packet collisions. Thus according to the ADR mechanism of LoRaWAN, nodes will try to improve reliability by using higher SFs, which consume more time on air, leading to higher energy consumption.

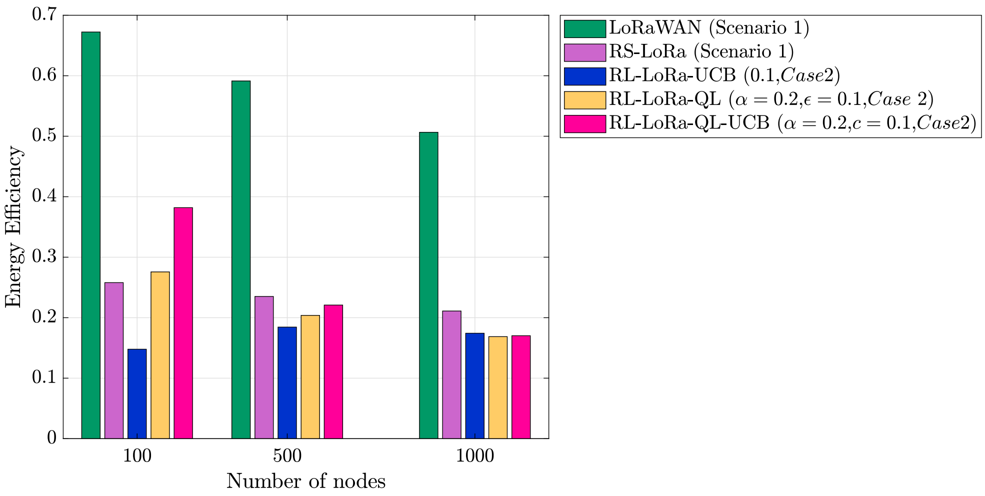

Figure 4.15: Energy efficiency under LoRaWAN, RS-LoRa, RL-LoRa.

From Fig. 4.15, it should also be noticed that the proposed RL-LoRa MAC protocol achieve a better energy efficiency than RS-LoRa only under small networks. The reason is that, in RS-LoRa, nodes are guided by the gateway to use a specific SF and transmit power, immediately the network is initiated [37]. In contrast, in RL-LoRa, nodes have to learn the best set of transmission parameters through interacting with the network. It will take time until the end devices "coordinate" their actions with each other (combinations of SF and transmit power) and find the transmission parameters that provide more reliable communication with the central gateway. Besides, the proposed RL algorithms only use two of the transmit powers allowed in LoRaWAN (in fact, the higher ones), while RS-LoRa exploits all the possibilities, enabling packets transmissions with very low powers [37].

To conclude, this section has compared the performance of the proposed RL-LoRa MAC protocol with the legacy LoRaWAN and RS-LoRa protocols. The simulation results demonstrated that RL-LoRa significantly improves the performance of both LoRaWAN and RS-LoRa in terms of the average PER and the network throughput. Although under the configurations used here, RL-LoRa presents a higher average network delay and energy consumption than LoRaWAN, it could be change by setting a lower T_{frame} and by defining actions than enable the RL algorithms to use all the transmit powers available in LoRaWAN, especially the lower ones.

The next section extends the performance comparison among LoRaWAN, RS-LoRa and the proposed RL-LoRa to scenarios with multiple gateways.

4.2.6 Performance comparison among the RL-LoRa, LoRaWAN and RS-LoRa protocols under multiple gateways scenarios

This section carries out a performance comparison among the proposed RL-LoRa (using the *RL-LoRa-QL* $-$ *UCB* algorithm set with optimized parameters) and the legacy LoRaWAN and RS-LoRa protocols. Specifically, the simulation results presented here, corresponds to a network configuration as that described for Scenario 2 in Chapter 3 (Section 3.2.3). Likewise, the average PER versus the distance to the gateway is used as a metric to compare the network reliability and scalability under the studied protocols. Table 4.8 summarizes the simulation setup for the protocols involved in the performance comparison presented in this section.

Table 4.8: Simulation setup for comparing the RL-LoRa, LoRaWAN, and RS-LoRa protocols under multiple gateways scenarios.

MAC protocol	**Simulation septup**
RL-LoRa	- *Case 2* and Scenario 2 - $T_{frame} = 120$ seconds - Exploration rate: $c = 0.1$ - Learning rate: $\alpha = 0.2$
LoRaWAN	- Scenario 2
RS-LoRa	- Scenario 2 - $T_{frame} = 60$ seconds

Fig. 4.16 represents the average PER versus distance to the central gateway under LoRaWAN, RS-LoRa, and the proposed RL-LoRa/*RL-LoRa-QL* $-$ *UCB* MAC protocol when considering deployments with multiple gateways. As can be observed, RL-LoRa/*RL-LoRa-QL* $-$ *UCB* outperforms LoRaWAN and RS-LoRa in terms of the average PER for all the distances. Most

importantly, it offers an average PER lower than 1% for all the distances and under all the considered number of nodes (100, 500, and 1000).

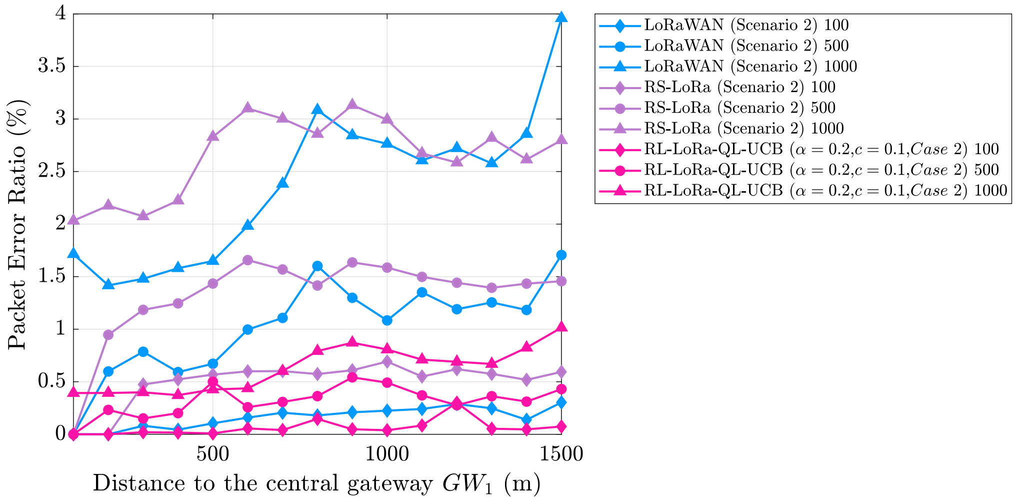

Figure 4.16: Average PER versus distance to the central gateway (GW_1) under the LoRaWAN, RS-LoRa, and RL-LoRa protocols

The evaluation results in Fig. 4.16, demonstrate that the proposed RL-LoRa MAC protocol can improve the LoRaWAN's performance not only under single gateway scenarios but also in deployments with multiple gateways. It is an expected result because under multiple gateways scenarios, the collisions that could not be avoided by the RL algorithms, it can still be solved by different gateways that can be reached.

4.2.7 Analyzing the proposed RL algorithms from the learning point of view

The last section of this chapter is dedicated to investigating the performance of the RL algorithms proposed in this thesis from the learning point of view. First, a general analysis about the "coordination" capabilities of the proposed *RL-LoRa-UCB*, *RL-LoRa-QL*, and *RL-LoRa-QL-UCB* algorithms is carried out. These algorithms will be compared in terms of the amount of time (time steps[11]) they require to "coordinate" the actions with the rest of active nodes (RL agents) in the network. Such amount of time is defined here as the "coordination time", which will be measured in terms of the time an RL-LoRa based network requires to stabilize the overall PDR when using a particular RL algorithm. After this general analysis, it will be investigated the convergence of the RL algorithm running at some specific nodes in the net-

[11] It should be noticed that here the steps of the RL algorithms correspond to the beginning of each frame of the RL-LoRa MAC protocols. That is, with the reception of a beacon message.

work. Two representative cases are chosen for this analysis: the RL algorithm running at the end devices with (1) the best and (2) the worst performance in terms of the average PER.

The simulations results presented through this section, correspond to the optimized parameters of the proposed *RL-LoRa-UCB*, *RL-LoRa-QL*, and *RL-LoRa-QL-UCB* algorithms ($c = 0.1$, $\alpha = 0.2$, $\epsilon = 0.1$). Likewise, they were obtained under Scenario 1's configuration (presented in Chapter 3, Section 3.2.3) and *Case 2* described in Section 4.2.4.2. Table 4.9 lists the simulation setup for each RL algorithm, as well as the distance from the considered nodes to the central gateway in all the cases.

Table 4.9: Simulation setup for analyzing the convergence of the proposed RL algorithms.

RL-LoRa/RL algorithm	**Simulation septup**
RL-LoRa/*RL-LoRa-UCB*	- *Case 2* and Scenario 1 - $T_{frame} = 120$ seconds - Exploration rate: $c = 0.1$ - Distance from nodes to the gateway: 84 m (best case) and 806 m (worst case)
RL-LoRa/*RL-LoRa-QL*	- *Case 2* and Scenario 1 - $T_{frame} = 120$ seconds - Learning rate: $\alpha = 0.2$ - Epsilon: $\epsilon = 0.1$ - Distance from nodes to the gateway: 136 m (best case) and 988 m (worst case)
RL-LoRa/$RL\text{-}LoRa\text{-}QL - UCB$	- *Case 2* and Scenario 1 - $T_{frame} = 120$ seconds - Exploration rate: $c = 0.1$ - Learning rate: $\alpha = 0.2$ - Distance from nodes to the gateway: 24 m (best case) and 971 m (worst case)

Fig. 4.17 shows the overall PDR versus the steps of all the proposed RL algorithms. As can be observed, among them, *RL-LoRa-QL-UCB* is the one with the lower "coordination time". That is, *RL-LoRa-QL-UCB* achieves a flat (no longer increases) overall PER curve earlier than the *RL-LoRa-UCB* and *RL-LoRa-QL* algorithms. It means that under RL-LoRa/*RL-LoRa-QL-UCB* all the active nodes can coordinate their actions in a few steps of the RL algorithm. They can find their best actions earlier and then ensure a high PDR in their data transmissions. For example, in a network with 1000 nodes, *RL-LoRa-QL-UCB* offers a coordination time of 50 steps (the PDR curve remains flat after approximately 50 steps), while *RL-LoRa-UCB* and *RL-LoRa-QL* presents a coordination time of around 100 and 200 steps. On this way, it is expected that RL-LoRa/*RL-LoRa-QL-UCB* provides a network performance

significantly higher than the offered by RL-LoRa/*RL-LoRa-UCB* and RL-LoRa/*RL-LoRa-QL*.

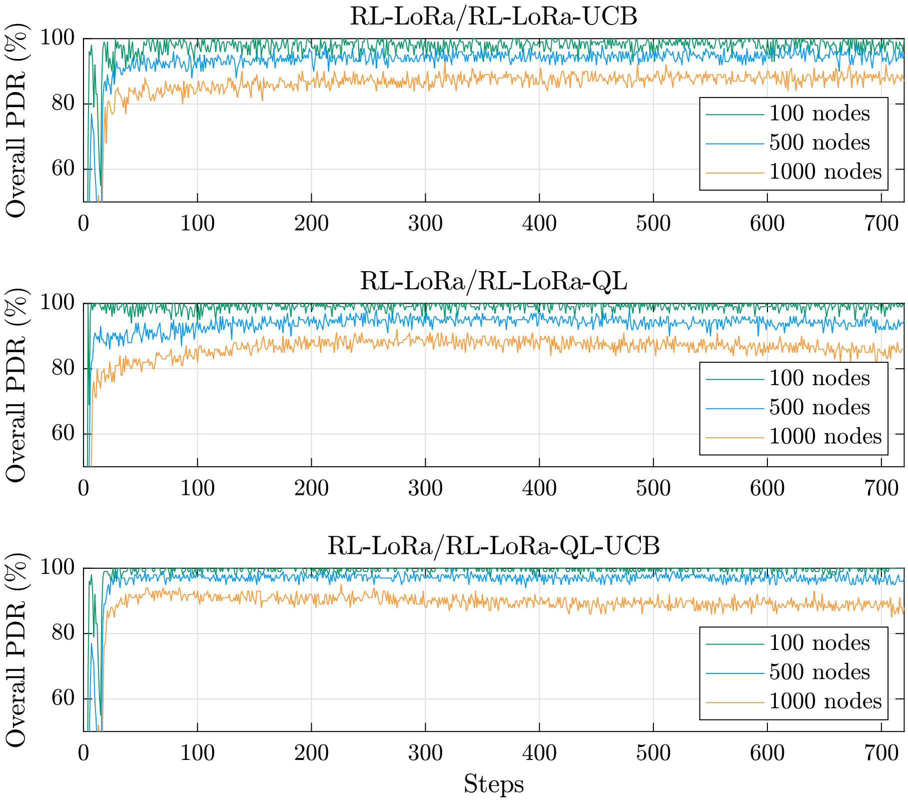

Figure 4.17: Average PDR versus steps of the RL algorithm under the proposed RL-LoRa/*RL-LoRa-UCB*, RL-LoRa/*RL-LoRa-QL*, and RL-LoRa/*RL-LoRa-QL-UCB* MAC protocols.

After comparing the coordination capabilities of the proposed *RL-LoRa-UCB*, *RL-LoRa-QL*, and *RL-LoRa-QL-UCB* algorithms, it is possible to compare them in terms of their convergence. In RL, the convergence of an RL algorithm can be demonstrated by plotting the learning curve (the average reward versus steps) and by showing the convergence of its action values. In practice, an RL algorithm is considered to converge when the learning curve gets flat and no longer increases. In terms of the action values, it is expected that they converge to its real values. That is, they remain constant over all the steps [71]. Since in the proposed system there is an independent RL algorithm (RL agent) running at each active node in the network, for this convergence study, two representative cases will be analyzed: the RL algorithms running (1) at nodes with an excellent performance in terms of the average PER (0%) and (2) at nodes with the worst performance (the highest average PER). The nodes under evaluation were selected from RL-LoRa networks with 500 nodes and are located at the distances shown in Table 4.9 from the central gateway. From these distances, it should be noticed how nodes close to the central gateway can experience 0% PERs, while the nodes with the worst performance are located at the cell edge. It demonstrates how the capture effect affects the performance of LoRaWAN networks, which was already discussed in Chapter 3.

Fig 4.18 shows the learning curve of the proposed RL algorithms when they are running at nodes under the two cases considered in this convergence analysis: nodes with the best and the worst performance in terms of the average PER. As can be observed, all the nodes with a perfect PER (0%) have the same learning curve, independently of the RL algorithm used. It is a constant with value '1' over all the steps of the considered RL algorithms. This is because the packets transmitted by these nodes (which are located very close to the central gateway) are never lost; thus, they receive always a reward of '1'. In contrast, the learning curve under the nodes with the worst PERs, have remarkable variations, especially during the first steps of the RL algorithms. It demonstrates that the RL algorithms running at the end devices with the worst performance (located far away from the central gateway), will take more time to achieve the desired convergence. However, it should be noticed that in this case, the variations on the average reward under *RL-LoRa-QL-UCB*, are smoother than the variations presented by the *RL-LoRa-UCB*, *RL-LoRa-QL* algorithms.

The figures 4.19 and 4.20 show the action values versus the steps of the RL algorithms proposed in this thesis, and under both the best and worse cases considered through this convergence analysis, respectively. As can be observed in Fig. 4.19, under the best case (nodes with a PER of 0% and that are close to the central gateway), *RL-LoRa-QL* is the algorithm with the highest variability of its action values, which are changing until the step 500, when the action a_2 remains as the action of highest value. This is because of the action selection rule used by *RL-LoRa-QL* (*ϵ-greedy*), which allows a random action selection with a probability ϵ even when the algorithm could have already found its best action. In contrast, it should be noticed how the *RL-LoRa-UCB* and *RL-LoRa-QL-UCB* algorithms can find their highest value action in few steps (around 15 steps), and remain using this action through all the subsequent steps. In this way, Fig. 4.19 demonstrates the strengths of the UCB's action selection rule over the *ϵ-greedy* approach, which was already discussed in this chapter when designing the proposed RL algorithms.

On the other hand, Fig. 4.20 shows that under the worse case (nodes with the worst PER in their networks and located far away from the central gateway), all the proposed RL algorithms present a significant variability in

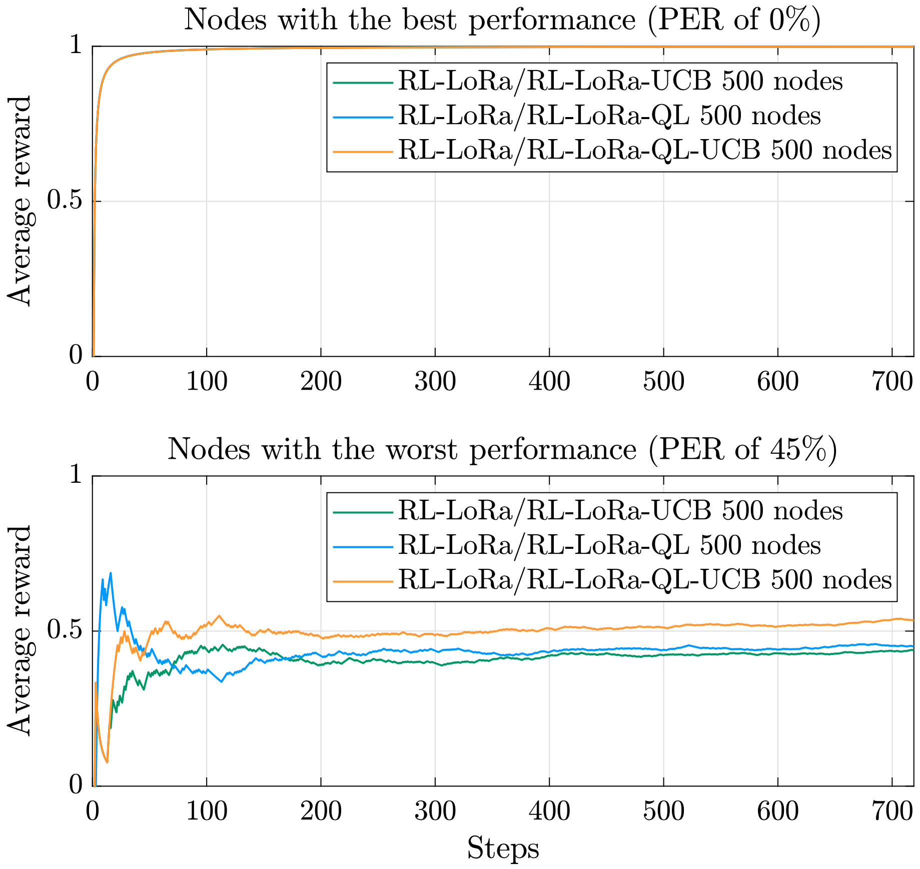

Figure 4.18: Average reward versus steps of the proposed *RL-LoRa-UCB*, *RL-LoRa-QL*, and *RL-LoRa-QL-UCB* algorithms under a network with 500 nodes.

their action values. This is because the performance of these nodes is seriously affected by the capture effect: their packets will arrive at the gateway with a very low power (due to the propagation losses) and will be probably dropped for not having the SNR required for their correct demodulation. Thus the RL algorithms running at such nodes will find it difficult to discover the best action (transmission parameters) to transmit packets because, probably, all the actions they select will receive a bad reward as a consequence of the high packet loss probability for these nodes. From the learning point of view, it results in the high variability of the action values of the RL algorithms being executed at these nodes, as shown in Fig. 4.20. However, it should be noticed that among the proposed RL algorithms, *RL-LoRa-QL* is the one that can find its best action earlier (the action a_{10}), demonstrating its superiority over the *RL-LoRa-UCB* and *RL-LoRa-QL* algorithms in the task of finding the best set of transmission parameters at the end devices, even in this extreme case.

To conclude, this chapter has introduced and evaluated the novel RL-

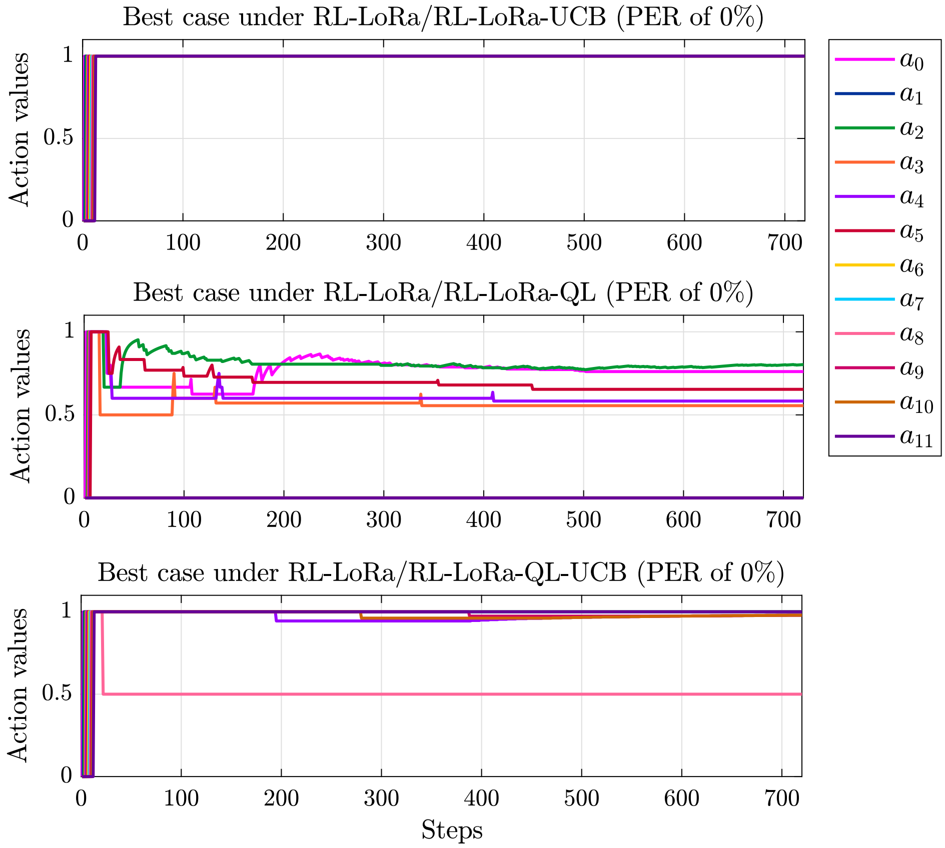

Figure 4.19: Action values versus steps of the proposed *RL-LoRa-UCB*, *RL-LoRa-QL*, and *RL-LoRa-QL-UCB* algorithms under the best case (nodes with a PER of 0%).

LoRa protocol, which has been designed to overcome LoRaWAN's performance in the task of controlling the MAC layer of LoRa based networks. The keys to the success of RL-LoRa are the use of RL algorithms for adjusting the transmission parameters at the end devices and the use of beacons for carrying the feedback required for such algorithms to rate the decisions (transmission parameters) they make. Three different algorithms were proposed in this chapter for locally adapting the transmission parameters at the end devices under the RL-LoRa protocol: the *RL-LoRa-UCB*, *RL-LoRa-QL*, and *RL-LoRa-QL-UCB* algorithms. The evaluation results have demonstrated that RL-LoRa significantly improves the performance of LoRaWAN and state-of-the-art solutions (the RS-LoRa protocol) under all the proposed RL algorithms. The best configuration of RL-LoRa (using the *RL-LoRa-QL-UCB* algorithm with its best parameter setting), can considerably reduce the average PER of LoRaWAN and RS-LoRa, by 15 and 10 %, respectively. This decrease in the PERs improves network reliability, which will further increase scalability. To the best of our knowledge, RL-LoRa is the first MAC protocol built upon the

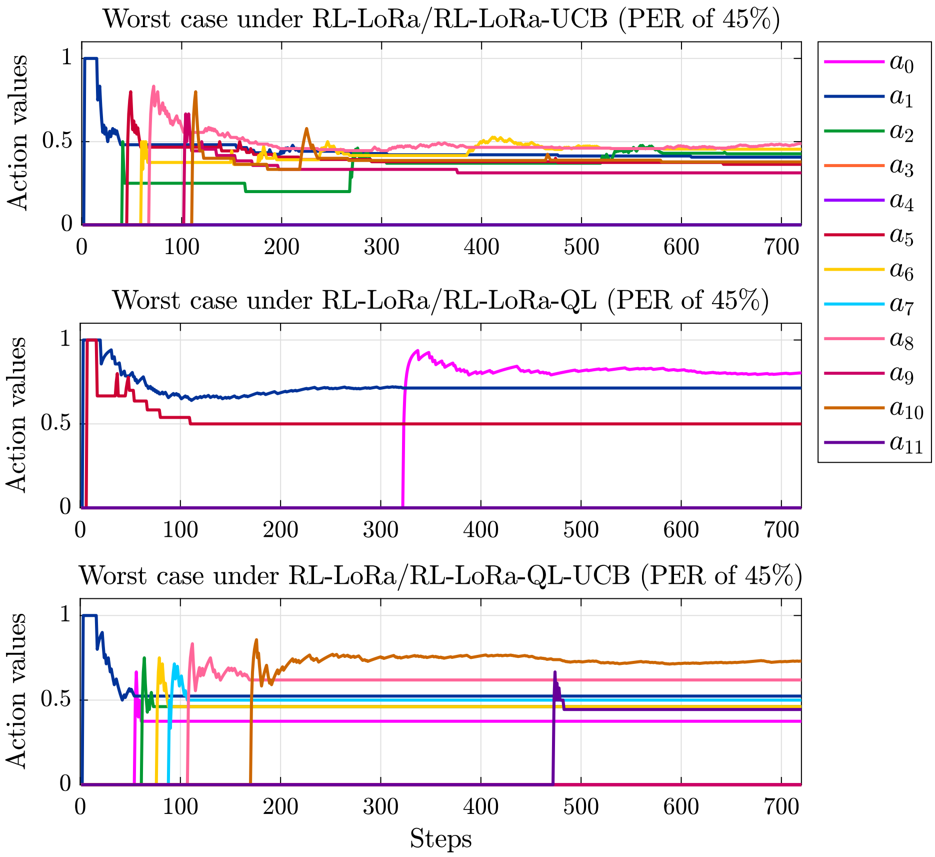

Figure 4.20: Action values versus steps of the proposed *RL-LoRa-UCB*, *RL-LoRa-QL*, and *RL-LoRa-QL-UCB* algorithms under the worst case (nodes with a PER of 45%).

LoRaWAN MAC layer that targets at improving the latter's reliability and scalability by applying RL techniques for locally adjusting the transmission parameters at the end devices.

5
Conclusions and Future Work

This final chapter summarizes the main conclusions from this thesis and discusses some future work and research directions.

5.1
Conclusions

The emergence of the LPWANs has created a novel research area that has been gaining popularity throughout the past few years. Due to the novelty of this field, many questions related to the performance of the existing LPWAN solutions and the possibilities they offer for improvements have arisen. On the other hand, the application of ML techniques to solve problems related to communication networks is another topic that has recently caught the research community's attention. In that sense, this thesis has taken the opportunity of working in two novel and prominent research areas: it has applied ML techniques to improve the performance of LoRaWAN, one of the existing LPWAN solutions.

The research tasks defined to accomplish the objectives of this thesis were completed at different phases, which are listed as follows. First, it was described three of the most well-known solutions for LPWANs: LoRaWAN, SigFox, and NB-IoT, highlighting LoRaWAN by its open-source nature and the flexibility of the modulation scheme it is based on (LoRa modulation), which open interesting opportunities for improvements. Then, it was summarized the fundamental concepts of RL, deepening on the UCB and Q-Learning algorithms, which have been widely applied to address common problems of wireless communication networks. After the introduction to the LPWANs and the fundamental concepts of RL, this thesis focused on identifying the limitations of the LoRaWAN MAC protocol through its evaluation, and on proposing RL based solutions to improve LoRaWAN's performance in terms of reliability and scalability. Finally, the proposed solutions were implemented in NS-3, which allowed assessing their performance and compared them with LoRaWAN and state-of-the-art solutions. The main conclusions from each of the phases previously mentioned are highlighted below.

Introduction to the existing LPWANs: The LPWAN solutions described in this thesis (LoRaWAN, SigFox, and NB-IoT) share common characteristics: they offer long-range communications and low energy consumption at the expense of low data rates and high latency. Although they try to meet the requirements of a variety of IoT application, they still face several challenges in terms of reliability and scalability. Among the existing solutions, LoRaWAN raises the most opportunities for performance improvements. This is because it is open-source and uses a flexible modulation scheme (LoRa modulation) that allows the SFs and transmit power adaptation.

Machine Learning for communication networks: Although all the ML paradigms (SL, USL, and RL) have been applied to fundamental problems in networking, including traffic classification, routing, resource management, congestion control, and network security, among others, RL results the most appropriate for solving problems that involve decision-making tasks, facilitating network scheduling, parameter adaptation, and resource allocation maximization under dynamic environments. For the above, this thesis selected RL as the main tool for improving the performance of LoRaWAN.

Analysis of the LoRaWAN protocol: The evaluation results presented in this thesis have shown that LoRaWAN faces several challenges in terms of reliability and scalability. Specifically, they have demonstrated that the average PER under LoRaWAN rapidly increases with the number of nodes in the network, exceeding the 30% at the cell edge for a network with 1000 nodes. The reason for these high PERs is the design of the LoRaWAN MAC layer. Although it offers the ADR mechanism for adjusting the transmission parameters at the end devices, this approach is not enough under dense networks. In such conditions, LoRaWAN's ADR mechanism results in an unfair network with high PERs for nodes far from the central gateway. These nodes experience significantly more collisions due to the capture effect. Thus, according to the ADR mechanism, the increase their SFs to improve reliability, but as a consequence only increase the number of collisions in the network. The evaluation results of LoRaWAN have also demonstrated that, for small networks (100 nodes), by using confirmed traffic, it is possible to achieve an average PER of 0% at the expense of higher energy consumption and network delay. However, this approach does not work for larger networks (with 500 and 1000 nodes), where the interference generated by the ACK transmissions leads to an average PER higher than 80%. In this context, the best solution for enabling a more reliable LoRaWAN network is deploying multiple gateways. The results presented in this work have shown that, under multiple gateway scenarios (seven enabled gateways), LoRaWAN can provide very low PERs, specifically, an average PER of 3% for networks with 1000 nodes. However, this solution for improving network reliability also involves a trade-off. In this case, it is between reliability and the cost of the network infrastructure due to the additional gateways that should be enabled. The main conclusion from the analysis of the LoRaWAN protocol is that under its simplified model, it can not guarantee reliability. However, it could be possible to provide a more reliable network by taking advantage of the modulation technique of LoRaWAN and proposing efficient methods to properly adjust the transmission parameters at the LoRaWAN end devices.

Improving LoRaWAN's performance through RL techniques: This thesis has proved that it is possible to considerably improve LoRaWAN's reliability and scalability by applying RL techniques. It proposed an extension of the legacy LoRaWAN MAC protocol that allows nodes to configure their transmission parameters themselves through RL algorithms. This extension, named RL-LoRa, provides an efficient mechanism to carry from gateways to the RL algorithms that are running at nodes, the reward information they require for rating their decisions. Three different RL algorithms were proposed in this thesis to operate in conjunction with RL-LoRa and to locally adjust the nodes' transmission parameters: *RL-LoRa-UCB*, *RL-LoRa-QL*, and *RL-LoRa-QL-UCB*. The simulation results have shown that under all of these algorithms, RL-LoRa provides significant performance improvements in both single and multiple gateways scenarios when compared with LoRaWAN. Specifically, RL-LoRa/*RL-LoRa-QL-UCB*, under its best setting of parameters ($c = 0.1$ and $\alpha = 0.2$), can reduce LoRaWAN's average PER in a 15% for networks with 1000 nodes. The price for these performance improvements is higher energy consumption and network delay. However, these metrics could be reduced by adjusting the frame duration of the RL-LoRa MAC protocol (T_{frame}) and using lower transmit powers when defining the actions used by the proposed RL algorithms.

Overall conclusion: This thesis has extended the knowledge about the existing LPWANs, more specifically, about the LoRaWAN protocol. The evaluation results have demonstrated that it is impossible to expect high reliability and scalability from LoRaWAN networks since its MAC protocol can not efficiently adjust the nodes' transmission parameters (SFs and transmit power) under dense deployments. However, this work has demonstrated that using RL techniques to locally control the transmission parameters at the end devices, is a prominent solution to provide reliable communication in LoRaWAN networks. It can significantly decrease the packet losses increasing the network reliability, which can further improve the network scalability.

5.2 Future work

Below, some future works related to this thesis are discussed.

5.2.1 Considering overall metrics when defining the reward signals

The RL-LoRa MAC protocol proposed in this thesis can improve LoRaWAN's reliability and scalability by using RL algorithms to control the transmission parameters at the end devices. However, it should be noticed that the RL-LoRa nodes are selfish on the decisions they make and, more importantly, they do not care about the influence of their decisions on the overall network performance. That is, in an RL-LoRa network, each node tries to minimize its packet losses, and, as a consequence, it contributes to decreasing the overall PER. In that sense, it could be interesting to make the RL algorithms running at nodes aware of how their decisions affect the overall network performance. One way of doing it is by defining the reward signal as a function of an overall metric (i.e., the average PER or the average PDR). For example, instead of rating the nodes' decisions (the selected transmission parameters) with just '1' if the previously transmitted packet was received at the gateway, or '0' if not, the reward signal could be defined as a weighted sum, where one of the terms could represent the overall PDR experienced as a consequence of the decision made. In this way, nodes will learn not just the action that improves their performance but also that contributes to increasing the overall PDR. It will especially help to improve the performance of nodes that are at the cell edge, thus alleviating the capture effect.

5.2.2 Improving the energy efficiency of RL-LoRa

Although RL-LoRa significantly improves the network performance in terms of the average PER when compared with LoRaWAN, it also offers worse energy efficiency. It could be possible to improve RL-LoRa's energy efficiency by considering more power levels (especially lower ones) when defining the actions used by the RL algorithms controlling the transmission parameters at the end devices. However, an interesting solution could be to define the reward signal for each node as a function of both the reliability and energy efficiency experienced as a consequence of its decisions. It will make the RL-LoRa nodes aware of the influence of their decisions on energy efficiency and, thus, they could learn the best action for both reducing their packet losses and ensuring high energy efficiency.

5.2.3 Coexistence analysis

All the evaluations carried out in this thesis have considered channels with just LoRaWAN communications on them. Since LoRaWAN operates on the ISM band, as future work, it could be investigated the effect of having different protocols working on the same band, on the performance of the proposed RL-LoRa protocol. It will probably increase the average PER under RL-LoRa, imposing new challenges to the approach of controlling the transmission parameters at LoRaWAN end devices through RL techniques.

Bibliography

[1] **LoRaWAN: What is it?.A technical overview of LoRa and LoRaWAN**, Nov. 2015. https://lora-alliance.org/sites/default/files/2018-04/what-is-lorawan.pdf.

[2] SUTTON, R. S.; BARTO, A. G.. **Reinforcement learning: an introduction**. The MIT Press, Cambridge, Massachusetts, second edition, 2018.

[3] LANSDOWNE, R.. **LoRaWAN Network Capacity Fundamentals**. https://privatevideos.hubs.vidyard.com/.

[4] SORNIN, N.; YEGIN, A.. **LoRaWAN Specification v1.0.3 | LoRa Alliance**, July 2018. https://lora-alliance.org/resource-hub/lorawanr-specification-v103.

[5] MEKKI, K.; BAJIC, E.; CHAXEL, F. ; MEYER, F.. **A comparative study of LPWAN technologies for large-scale IoT deployment**. ICT Express, 5(1):1–7, Mar. 2019.

[6] QADIR, Q. M.; RASHID, T. A.; AL-SALIHI, N. K.; ISMAEL, B.; KIST, A. A. ; ZHANG, Z.. **Low power wide area networks: A survey of enabling technologies, applications and interoperability needs**. IEEE Access, 6:77454–77473, 2018.

[7] RAZA, U.; KULKARNI, P. ; SOORIYABANDARA, M.. **Low power wide area networks: An overview**. IEEE Communications Surveys and Tutorials, 19(2):855–873, 2017.

[8] **Sigfox - The Global Communications Service Provider for the Internet of Things (IoT)**. `http://www.sigfox.com`, accessed: 2019-11-27.

[9] **Semtech LoRa Technology Overview | Semtech**. `https://www.semtech.com/lora`, accessed: 2019-11-27.

[10] POPLI, S.; JHA, R. K. ; JAIN, S.. **A Survey on Energy Efficient Narrowband Internet of Things (NBIoT): Architecture, Application and Challenges**. 7:16739–16776, 2019.

[11] LAVRIC, A.; PETRARIU, A. I. ; POPA, V.. **Long Range SigFox Communication Protocol Scalability Analysis Under Large-Scale, High-Density Conditions**. IEEE Access, 7:35816–35825, 2019.

[12] LABS, L.. **Symphony Link - LPWAN Protocol Data Sheet**. https://www.link-labs.com/symphony-link-data-sheet.

[13] HASSAN, G.; HASSANEIN, H. S.. **MoT: A Deterministic Latency MAC Protocol for Mission-Critical IoT Applications**. In: 2018 14TH International Wireless Communications Mobile Computing Conference (IWCMC), p. 588–593, June 2018. ISSN: 2376-6506.

[14] BALEVI, E.; RABEE, F. T. A. ; GITLIN, R. D.. **ALOHA-NOMA for Massive Machine-to-Machine IoT Communication**. In: 2018 IEEE International Conference on Communications (ICC), p. 1–5, May 2018. ISSN: 1938-1883.

[15] **LoRaWAN Coverage | LoRa Alliance**. https://lora-alliance.org/lorawan-coverage.

[16] **LoRaWAN Vertical Markets | LoRa Alliance**. https://lora-alliance.org/lorawan-vertical-markets.

[17] REYNDERS, B.; WANG, Q. ; POLLIN, S.. **A LoRaWAN module for ns-3: implementation and evaluation**. p. 61–68, Surathkal, India, 2018. ACM Press.

[18] **LoRaWAN for Developers | LoRa Alliance**. https://lora-alliance.org/lorawan-for-developers.

[19] **AN1200.22 LoRaTM Modulation Basics**, May 2015.

[20] REYNDERS, B.; MEERT, W. ; POLLIN, S.. **Power and spreading factor control in low power wide area networks**. In: 2017 IEEE International Conference ON Communications (ICC), p. 1–6, Paris, France, May 2017. IEEE.

[21] ADELANTADO, F.; VILAJOSANA, X.; TUSET-PEIRO, P.; MARTINEZ, B.; MELIA-SEGUI, J. ; WATTEYNE, T.. **Understanding the Limits of LoRaWAN**. IEEE Communications Magazine, 55:34–40, Sept. 2017.

[22] GEORGIOU, O.; RAZA, U.. **Low Power Wide Area Network Analysis: Can LoRa Scale?** IEEE Wireless Communications Letters, 6(2):162–165, Apr. 2017.

[23] MIKHAYLOV, K.; PETÄJÄJÄRVI, J. ; JANHUNEN, J.. **On LoRaWAN scalability: Empirical evaluation of susceptibility to inter-network interference**. In: 2017 European Conference on Networks and Communications (EuCNC), p. 1–6, June 2017.

[24] JETMIR HAXHIBEQIRI; FLORIS VAN DEN ABEELE; INGRID MOERMAN ; JEROEN HOEBEKE. **LoRa Scalability: A Simulation Model Based on Interference Measurements**. Sensors, 17(6):1193, May 2017.

[25] VAN DEN ABEELE, F.; HAXHIBEQIRI, J.; MOERMAN, I. ; HOEBEKE, J.. **Scalability Analysis of Large-Scale LoRaWAN Networks in ns-3**. IEEE Internet of Things Journal, 4(6):2186–2198, Dec. 2017.

[26] PETÄJÄJÄRVI, J.; MIKHAYLOV, K.; PETTISSALO, M.; JANHUNEN, J. ; IINATTI, J.. **Performance of a low-power wide-area network based on LoRa technology: Doppler robustness, scalability, and coverage**. International Journal of Distributed Sensor Networks, 13(3), Mar. 2017.

[27] SANCHEZ-IBORRA, R.; SANCHEZ-GOMEZ, J.; BALLESTA-VIÑAS, J.; CANO, M.-D. ; SKARMETA, A. F.. **Performance Evaluation of LoRa Considering Scenario Conditions**. Sensors, 18(3), Mar. 2018.

[28] FRÓMETA FONSECA, D.; C. DE LAMARE, R. ; L. MADRUGA, E.. **Performance Evaluation of LoRaWAN and RS-LoRa in Mobile Scenarios**. Petrópolis, Rio de Janeiro, 2019.

[29] HAXHIBEQIRI, J.; DE POORTER, E.; MOERMAN, I. ; HOEBEKE, J.. **A Survey of LoRaWAN for IoT: From Technology to Application**. Sensors, 18(11):3995, Nov. 2018.

[30] BOR, M.; ROEDIG, U.. **LoRa Transmission Parameter Selection**. In: 2017 13TH International Conference on Distributed Computing in Sensor Systems (DCOSS), p. 27–34, June 2017. ISSN: 2325-2944.

[31] TIURLIKOVA, A.; STEPANOV, N. ; MIKHAYLOV, K.. **Method of Assigning Spreading Factor to Improve the Scalability of the LoRaWan Wide Area Network**. In: 2018 10TH International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), p. 1–4, Moscow, Russia, Nov. 2018. IEEE.

[32] EL-AASSER, M.; ELSHABRAWY, T. ; ASHOUR, M.. **Joint Spreading Factor and Coding Rate Assignment in LoRaWAN Networks**.

In: 2018 IEEE Global Conference on Internet of Things (GCIoT), p. 1–7, Alexandria, Egypt, Dec. 2018. IEEE.

[33] ABDELFADEEL, K. Q.; CIONCA, V. ; PESCH, D.. **Fair Adaptive Data Rate Allocation and Power Control in LoRaWAN**. In: 2018 IEEE 19TH International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM), p. 14–15, June 2018. ISSN: null.

[34] CUOMO, F.; CAMPO, M.; CAPONI, A.; BIANCHI, G.; ROSSINI, G. ; PISANI, P.. **EXPLoRa: Extending the performance of LoRa by suitable spreading factor allocations**. In: 2017 IEEE 13TH International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), p. 1–8, Oct. 2017. ISSN: null.

[35] SLABICKI, M.; PREMSANKAR, G. ; DI FRANCESCO, M.. **Adaptive configuration of lora networks for dense IoT deployments**. In: NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium, p. 1–9, Apr. 2018. ISSN: 2374-9709.

[36] LI, S.; RAZA, U. ; KHAN, A.. **How Agile is the Adaptive Data Rate Mechanism of LoRaWAN?** In: 2018 IEEE Global Communications Conference (GLOBECOM), p. 206–212, Dec. 2018. ISSN: 1930-529X.

[37] REYNDERS, B.; WANG, Q.; TUSET-PEIRO, P.; VILAJOSANA, X. ; POLLIN, S.. **Improving Reliability and Scalability of LoRaWANs Through Lightweight Scheduling**. IEEE Internet of Things Journal, 5(3):1830–1842, June 2018.

[38] OH, Y.; LEE, J. ; KIM, C.-K.. **TRILO: A Traffic Indication-Based Downlink Communication Protocol for LoRaWAN**. Wireless Communications and Mobile Computing, 2018:1–14, Sept. 2018.

[39] RIZZI, M.; DEPARI, A.; FERRARI, P.; FLAMMINI, A.; RINALDI, S. ; SISINNI, E.. **Synchronization Uncertainty Versus Power Efficiency in LoRaWAN Networks**. IEEE Transactions on Instrumentation and Measurement, 68(4):1101–1111, Apr. 2019.

[40] POLONELLI, T.; BRUNELLI, D. ; BENINI, L.. **Slotted ALOHA Overlay on LoRaWAN - A Distributed Synchronization Approach**. In: 2018 IEEE 16TH International Conference on Embedded and Ubiquitous Computing (EUC), p. 129–132, Oct. 2018. ISSN: null.

[41] HAXHIBEQIRI, J.; MOERMAN, I. ; HOEBEKE, J.. **Low Overhead Scheduling of LoRa Transmissions for Improved Scalability**. IEEE Internet of Things Journal, 6(2):3097–3109, Apr. 2019.

[42] AZARI, A.; CAVDAR, C.. **Self-Organized Low-Power IoT Networks: A Distributed Learning Approach**. In: 2018 IEEE Global Communications Conference (GLOBECOM), p. 1–7, Dec. 2018. ISSN: 1930-529X.

[43] KERKOUCHE, R.; ALAMI, R.; FÉRAUD, R.; VARSIER, N. ; MAILLÉ, P.. **Node-based optimization of LoRa transmissions with Multi-Armed Bandit algorithms**. In: 2018 25TH International Conference on Telecommunications (ICT), p. 521–526, June 2018.

[44] WANG, M.; CUI, Y.; WANG, X.; XIAO, S. ; JIANG, J.. **Machine Learning for Networking: Workflow, Advances and Opportunities**. IEEE Network, 32(2):92–99, Mar. 2018.

[45] **LoRa Alliance**. https://lora-alliance.org/.

[46] GOURSAUD, C.; MO, Y.. **Random unslotted time-frequency ALOHA: Theory and application to IoT UNB networks**. In: 2016 23RD International Conference on Telecommunications (ICT), p. 1–5, May 2016.

[47] BOUTABA, R.; SALAHUDDIN, M. A.; LIMAM, N.; AYOUBI, S.; SHAHRIAR, N.; ESTRADA-SOLANO, F. ; CAICEDO, O. M.. **A comprehensive survey on machine learning for networking: evolution, applications and research opportunities**. Journal of Internet Services and Applications, 9(1):16, Dec. 2018.

[48] ALSHEIKH, M. A.; LIN, S.; NIYATO, D. ; TAN, H.-P.. **Machine Learning in Wireless Sensor Networks: Algorithms, Strategies, and Applications**. IEEE Communications Surveys & Tutorials, 16(4):1996–2018, 2014.

[49] ZHENZHEN LIU; ELHANANY, I.. **RL-MAC: A QoS-Aware Reinforcement Learning based MAC Protocol for Wireless Sensor Networks**. In: 2006 IEEE International Conference on Networking, Sensing and Control, p. 768–773. IEEE, 2006.

[50] CHU, Y.; MITCHELL, P. D. ; GRACE, D.. **ALOHA and Q-Learning based medium access control for Wireless Sensor Networks**. In: 2012 International Symposium on Wireless Communication Systems (ISWCS), p. 511–515, 2012.

[51] KOSUNALP, S.; CHU, Y.; MITCHELL, P. D.; GRACE, D. ; CLARKE, T.. **Use of Q-learning approaches for practical medium access control in wireless sensor networks**. Engineering Applications of Artificial Intelligence, 55:146–154, Oct. 2016.

[52] SAVAGLIO, C.; PACE, P.; ALOI, G.; LIOTTA, A. ; FORTINO, G.. **Lightweight Reinforcement Learning for Energy Efficient Communications in Wireless Sensor Networks**. IEEE Access, 7:29355–29364, 2019.

[53] NIU, J.; DENG, Z.. **Distributed self-learning scheduling approach for wireless sensor network**. Ad Hoc Networks, 11(4):1276–1286, June 2013.

[54] BONNEFOI, R.; MOY, C. ; PALICOT, J.. **Improvement of the LPWAN AMI backhaul's latency thanks to reinforcement learning algorithms**. EURASIP Journal on Wireless Communications and Networking, 2018(1):34, Dec. 2018.

[55] JIANG, N.; DENG, Y.; NALLANATHAN, A. ; CHAMBERS, J. A.. **Reinforcement Learning for Real-Time Optimization in NB-IoT Networks**. IEEE Journal on Selected Areas in Communications, 37(6):1424–1440, June 2019.

[56] SZEPESVÁRI, C.. **Algorithms for Reinforcement Learning**. Synthesis Lectures on Artificial Intelligence and Machine Learning, 4(1):1–103, Jan. 2010.

[57] RAVICHANDIRAN, S.. **Hands-On Reinforcement Learning with Python: Master reinforcement and deep reinforcement learning using OpenAI Gym and TensorFlow**. Packt Publishing Ltd, June 2018.

[58] WATKINS, C. J. C. H.. **Learning from Delayed Rewards**. Cambridge University, 1989.

[59] Wiering, M.; Otterlo, M. v., editors. **Reinforcement Learning: State-of-the-Art**. Adaptation, Learning, and Optimization. Springer-Verlag, Berlin Heidelberg, 2012.

[60] CLAUS, C.; BOUTILIER, C.. **The Dynamics of Reinforcement Learning in Cooperative Multiagent Systems**. p. 7.

[61] **SX1272 | Long Range, Low Power RF Transceiver 860-1000mhz | Semtech**. https://www.semtech.com/products/wireless-rf/lora-transceivers/sx1272.

[62] **SX1301 | Digital Baseband Chip LoRaWAN macro gateways | Semtech.** https://www.semtech.com/products/wireless-rf/lora-gateways/sx1301.

[63] BOR, M. C.; ROEDIG, U.; VOIGT, T. ; ALONSO, J. M.. **Do LoRa Low-Power Wide-Area Networks Scale?** In: PROCEEDINGS OF THE 19TH ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems - MSWiM '16, p. 59–67, Malta, Malta, 2016. ACM Press.

[64] POP, A.-I.; RAZA, U.; KULKARNI, P. ; SOORIYABANDARA, M.. **Does Bidirectional Traffic Do More Harm Than Good in LoRaWAN Based LPWA Networks?** In: GLOBECOM 2017 - 2017 IEEE Global Communications Conference, p. 1–6, Dec. 2017.

[65] SLABICKI, M.; PREMSANKAR, G. ; DI FRANCESCO, M.. **Adaptive configuration of lora networks for dense IoT deployments.** In: NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium, p. 1–9, Apr. 2018. ISSN: 2374-9709.

[66] NSNAM. **ns-3.** https://www.nsnam.org/.

[67] MAGRIN, D.; CENTENARO, M. ; VANGELISTA, L.. **Performance evaluation of LoRa networks in a smart city scenario.** In: 2017 IEEE International Conference on Communications (ICC), p. 1–7, May 2017. ISSN: 1938-1883.

[68] KRITSIS, K.; PAPADOPOULOS, G. Z.; GALLAIS, A.; CHATZIMISIOS, P. ; THÉOLEYRE, F.. **A Tutorial on Performance Evaluation and Validation Methodology for Low-Power and Lossy Networks.** IEEE Communications Surveys Tutorials, 20(3):1799–1825, 2018.

[69] JAIN, R.; CHIU, D.-M. ; HAWE, W.. **A Quantitative Measure Of Fairness And Discrimination For Resource Allocation In Shared Computer Systems.** ArXiv, cs.NI/9809099, 1998.

[70] AUER, P.; CESA-BIANCHI, N.; FREUND, Y. ; SCHAPIRE, R. E.. **The Nonstochastic Multiarmed Bandit Problem.** SIAM Journal on Computing, 32(1):48–77, Jan. 2003.

[71] CLAUS, C.; BOUTILIER, C.. **The Dynamics of Reinforcement Learning in Cooperative Multiagent Systems.** American Association for Artificial Intelligence (AAAI), p. 7, 1998.

[72] BONNEFOI, R.; BESSON, L.; MOY, C.; KAUFMANN, E. ; PALICOT, J.. **Multi-Armed Bandit Learning in IoT Networks: Learning helps even in non-stationary settings**. arXiv:1807.00491 [cs], July 2018. arXiv: 1807.00491.

[73] REYNDERS, B.; POLLIN, S.. **Chirp spread spectrum as a modulation technique for long range communication**. In: 2016 Symposium on Communications and Vehicular Technologies (SCVT), p. 1–5, Mons, Belgium, Nov. 2016. IEEE.

[74] FIALHO, V.; AZEVEDO, F.. **Wireless Communication Based on Chirp Signals for LoRa IoT Devices**. i-ETC : ISEL Academic Journal of Electronics Telecommunications and Computers, 4(1):6–6, Dec. 2018.

[75] SPRINGER, A.; GUGLER, W.; HUEMER, M.; REINDL, L.; RUPPEL, C. ; WEIGEL, R.. **Spread spectrum communications using chirp signals**. In: IEEE/AFCEA EUROCOMM 2000. Information Systems for Enhanced Public Safety and Security (Cat. No.00EX405), p. 166–170, May 2000.

[76] WANG, X.; LI, X. ; FEI, M.. **Performance evaluation of wireless networks based on chirp spread spectrum**. In: 2008 11TH IEEE International Conference on Communication Technology, p. 58–61, Nov. 2008.

A
Chirp Spread Spectrum Technique

CSS is a spread spectrum technique that uses chirp signals to transmit data. It defines the following relation between the bit rate R_b, bandwidth B, spreading factor SF and symbol rate R_s [19]:

$$R_b = \frac{B \times SF}{2^{SF}} = R_s \times SF \quad [bits/sec]. \tag{A-1}$$

Similarly, the symbol period T_s, defined as the reciprocal of R_s, is given by:

$$T_s = \frac{1}{R_s} = \frac{2^{SF}}{B} \quad [secs]. \tag{A-2}$$

As depicted in equations (A-1) and (A-2), a lower SF leads to a higher data rate and shorter transmission time. However, it also requieres a higher SNR at the receiver to correctly demodulate the signals [61] and corresponds to the less robust symbol.

There are four important parameters related to the CSS symbol generation: the SF, the minimal frequency f_{min}, the maximal frequency f_{max}, and a starting frequency f_0 [73]. The SF corresponds to the number of bits per symbol, which is translated into a starting frequency f_0 that encodes the information of the symbol according to [74]:

$$f_0 = \frac{sv}{2^{SF}} \times T_s, \tag{A-3}$$

where sv corresponds to he symbol value (0 to 2^{SF} - 1) and T_s is given by Eq. (A-2).

In order to generate a CSS symbol, depending on the SF value and for an specific bandwidth ($B = f_{max}$ - f_{min}), the starting frequency f_0 is linearly increased to f_{max}, then continuing from f_{min} until f_0 is reached again. The mathematical representation of the corresponding chirp signal is given by:

$$s(t) = e^{j2\pi f(t)t}, \tag{A-4}$$

where $f(t)$ is the frequency in function of time [73]. As example, two CSS symbols with different SFs are shown in Fig. A.1, including all the parameters used for their generation.

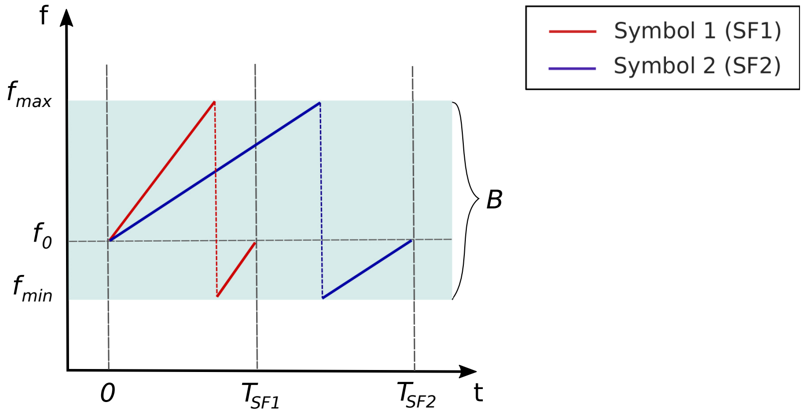

Figure A.1: Graphical representation of the frequency in function of time $f(t)$ for two CSS symbols with different SFs (SF1<SF2).

It should be noticed that, according to Eq. (A-3), with SF = 7 it is possible to encode 128 symbols lasting T_s seconds (according to Eq. (A-2)) within a specific bandwidth. Furthermore, thaks to the different SFs (data rates), different symbols can be demodulated simultaneously. On the other hand, because the generated CSS symbols are close to be orthogonal [73], a correlation based decoder can be used to decode the received signals. Every received symbol is correlated with the base CSS symbols, then the decoder makes a decision based on the maximum correlation [73].

Different expressions can be found in the literature to evaluate the performance of CSS under various SNR scenarios [73, 75, 76]. This thesis, uses the close-form equation for the Bit Error Rate of CSS symbols derived by authors in [73]:

$$P_{e,CSS} = \frac{1}{2}Q\Big(1.28\sqrt{SF\frac{E_b}{N_0}} - \sqrt{SF}1.28 + 0.4\Big), \tag{A-5}$$

where, E_b/N_0 is the energy per bit to noise density, which can be obtained from the SNR as follows:

$$\frac{E_b}{N_0} = \frac{P_s B}{P_n R_b(SF)}, \tag{A-6}$$

being P_s/P_n the SNR, and $R_b(SF)$ the bit rate associated to the used SF (Eq. (A-1)).

Finally, it is worth highlighting that the expression in Eq. (A-5) was deduced for a correlation-based detector with Additive White Gaussian Noise (AWGN).