Università degli Studi di Padova

Performance evaluation of a LoRaWAN network for confirmed and synchronous traffic

Department of Information Engineering

Thesis in Telecommunication Engineering

Supervisor
Michele Zorzi
UNIVERSITÀ DI PADOVA

Co-Supervisor
Davide Magrin
UNIVERSITÀ DI PADOVA

Candidate
Alicia Grande Gutiérrez

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

3GPP 3rd Generation Partnership Project

ACK Acknowledgement

ADR Adaptive Data Rate

CF Carrier Frequency

CSS Chirp Spread Spectrum

DL Downlink

ED End Device

EH Energy Harvesting

ETSI European Telecommunications Standard Institute

FED Forward Error Correction

FFT Fast Fourier Transform

GW Gateway

IEEE Institute of Electrical and Electronics Engineers

IoT Internet of Things

ISM Industrial, Scientific, and Medical

LAN Local Area Network

LPWAN Low-Power Wide Area Network

LR-WPAN Low-Rate Wireless Personal Area Network

LTE Long Term Evolution

LTN Low Throughput Networks

MAC Medium Access Control

MFSK Multiple Frequency Shift Keying

NS3 Network Simulator 3

NS Network Server

PHY Physical Layer

SF Spreading Factor

SISO Single Input Single Output

SWIPT Simultaneous Wireless Information and Power Transfer

UNB Ultra Narrow Band

WCDMA Wideband Code Division Multiple Access

WLAN Wireless Local Area Networks

UL Uplink

UNB Ultra Narrow Band

INTRODUCTION

Recent developments in the electronic device market and growth in the communication needs of the contemporary society are marking the beginning of a new era of Internet of Things (IoT). This technological revolution will enable real world, physical objects to connect to the internet, becoming "smart things", capable of sensing, actuating and coordinating with other devices.

Every day more and more devices are connected to the internet. This has led to whole new scenarios and to the need for greater network efficiency, in order to connect an ever-increasing amount of IoT devices. In this situation, new technical requirements appear, among which we can highlight the need for a longer battery life, better coverage, lower interference levels, lower device costs and support for a massive and growing number of devices.

In the first part of this thesis, we will perform some evaluations on the impact of confirmed traffic in one of the most prominent protocols for IoT, LoRaWAN. To do so, we will leverage a network simulator, ns-3, to understand how packets requiring an acknowledgement can hinder the performance of the whole network.

In the second part of the thesis, we will tackle one of the possible applications of the IoT paradigm: the implementation of alarm devices. Some examples of applications of this use case could be in house security, flooding prevention, smart network monitoring, prevention of forest fires. In a scenario where there is a network of devices that could potentially trigger simultaneously for transmission, different problems can arise, leading to the loss of the packets due to collisions of the packets sent from the alarm devices to the gateways that receives the information. In this case, we will use ns-3 to analyze the problem and propose some solutions.

1 PROTOCOLS FOR THE INTERNET OF THINGS

Conventionally, electronic devices have been connected by physical communication links. However, a cabled solution to a pervasive IoT network would entail installation inconveniencies that would make maintenance costs unbearable. For this reason, wireless technology is preferred to connect IoT devices, despite undesirable characteristics of the radio link such as varying propagation conditions, the presence of interference, increased security vulnerability and radio band regulations.

There are various IoT applications, such as smart metering, smart cities (parking sensors and waste management), smart buildings (home automation), process monitoring of industrial procedures, healthcare applications, environmental monitoring, etc. These applications require the underlying communication technologies to have the following features:

- **High autonomy of the devices**: some devices, especially those that are difficult to access, require a low power consumption to maximize battery life and thus decrease maintenance costs. IoT applications are currently limited by their finite battery life, which is in the order of 2-3 years. The new class of LPWANs is expected to extend the battery life of sensors in order to surpass the 10 year target [1], by employing smart modulation schemes. One of the possible additional solutions is foreseen to be energy harvesting (EH), a paradigm consisting in converting energy sources into electricity, and recharging batteries using the surrounding environment. Although this energy collection scheme could solve the energy problem, its main disadvantage consists in the fact that natural resources are uncontrollable. For this reason, alternative energy sources are being investigated, such as the technique known as Simultaneous Wireless Information and Power Transfer (SWIPT), which proposes harvesting energy from radio frequency signals.
- **Low cost of the devices and their deployment**: the total cost of devices should be as low as possible, so that they can be marketed

more easily. As a result, the current industry targets a module cost of less than 5$ [2].

- **Coverage**: IoT applications can require both external and internal coverage. Therefore, devices should not be affected, as much as possible, by obstacles in outdoor environments, as walls or even whole buildings.
- **Support for a massive number of devices**: base stations should be able to give support to large numbers pf devices that require simultaneous connection, with densities in the order of 10^4 devices in a 1 km radius. In this context, where many wireless technologies and massive number of devices are used, there is a high probability of interference, which causes a relevant degradation in the quality of the wireless link. This is the case of most LPWANs, because of the use of crowded ISM bands and of the ALOHA access scheme. Therefore, interference mitigation techniques become very relevant. Due to the high number of devices connected today, and knowing that this amount grows exponentially, it is also necessary to design transceivers which are reconfigurable, adapting to the requirements of the network and connected devices at every moment. Operation over multiple-frequency bands is also a measure that is foreseen to increase resistance to interference.
- **Security**: minimize the vulnerability of the cyber and physical components of the system, securing the information flow between the smart object and the final data user.

In the recent years, short-range wireless networks such as Bluetooth, ZigBee or Z-wave have been extensively used. Other options include local area networks (WLANs) such as WiFi and HiperLAN or cellular technologies such as the GSM and LTE networks. The following sections of this chapter briefly describe these technologies and their impact on the IoT paradigm, detailing how the issues of high costs, a high energy consumption and a high complexity have hindered their diffusion. Finally, the class of Low Power Wide Area Networks (LPWANs) will be introduced. A detailed description of the

LoRaWAN technology, which belongs to this class of networks and will be the main focus of this thesis, is described in detail in the next chapter.

1.1 Low-Rate Wireless Personal Area Networks (LR-WPAN)

LR-WPANs are a type of network designed for low cost and very low-power transmission for short-range wireless communications. These characteristics are different from what are common design targets for wireless technologies, like high throughput and low delay.

Wireless Local Area Networks (WLANs) have been created as a development of wired Local Area Networks (LANs). One of the main advantages is the decrement of the cost due to removal of the both wire and physical connection between nodes. Wireless Personal Area Networks (WPANs) have a personal coverage of 10 m in all directions for both the cases of the person moving or being stationary. WPANs allow transmission of information on a short distance between a small number of participants. The main advantage compared to WLAN is that WPAN networks need only a basic network infrastructure. This decreases the total cost of the system, at the same time that it reduces power consumption and the size of the implementations.

The IEEE 802.15 working group has defined three WPAN classes:

- The high data rate WPAN (IEEE 802.15.3) is designed for multimedia applications that need a high Quality of Service (QoS).
- Medium Speed WPAN (IEEE 802.15.1 / Bluetooth) is designed to replace wires in electronic devices such as mobile phones and PDAs, with a QoS suitable for voice applications.

  LR-WAN (IEEE 802.15.4) is designed for devices that require low power consumption, data rate and QoS.

The figure shows the operating space of the main IEEE 802.15 WPAN standards.

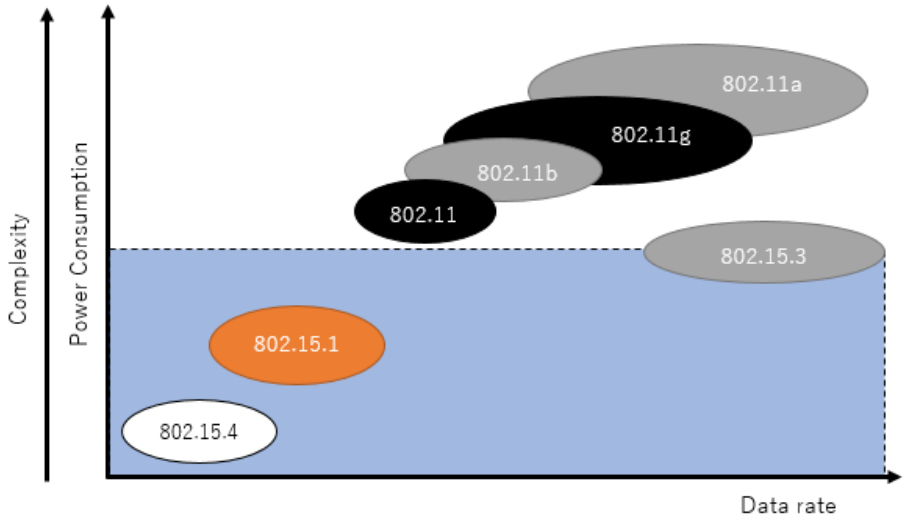

Figure 1-1 Operation space of different WPAN networks [3]

Most LR-WPANs operate in the unlicensed ISM bands. These are radio bands reserved internationally for industrial, scientific or medical purposes. More specifically, they operate in the bands centered on 2.4GHz, 868/915 MHz, 433 MHz, and 169 MHz, depending on the region of operation [3].

1.2 CELLULAR IOT

The 3rd Generation Partnership Project (3GPP) unites telecommunications standard development organizations and produces the Reports and Specifications that define 3GPP technologies. The project covers cellular telecommunications network technologies, including radio access, the core transport network, and service capabilities - including work on codecs, security, quality of service - and thus provides complete system specifications. [4]

Figure 1-2 3GPP

3GPP has developed several IoT communication models:

- LTE Cat 1 was the first LTE implementation, and although it was not developed for IoT applications, it can still be used for this use case. It was published in 2008.
- LTE Cat 0 is an improvement of the previous version. It was published in 2015.
- LTE Cat M1 (eMTC) was developed thinking about the functionalities that could be added to LTE for IoT and Machine to Machine (M2M) devices. It takes advantage of existing LTE base stations to provide extended coverage to IoT nodes, allowing high data speed in applications. It was published in 2016.
- LTE Cat NB1 (NB-IoT) was developed focusing on IoT and was also published in 2016.

Table 1 provides a quantitative comparison of the 3GPP standards for IoT:

	LTE Cat 1	***LTE Cat 2***	***LTE Cat M1***	***LTE Cat NB***
3GPP Release	8	12	13	13
Downlink peak Rate	10 Mbps	1Mbps	1Mbps	250 kpbs
Uplink Peak Rate	5 Mbps	1 Mbps	1 Mbps	250 kbps (Multi tone) / 20 kbps (Single tone)
Antennas	2	1	1	1
Receive Bnadwidth	1.08 -18 MHZ	1.08 -18 MHZ	18 MHZ	180 kHz

Table 1 Comparison of 3GPP models

1.2.1.1 NB-IOT

Narrowband IoT is the main emerging IoT technology set up by 3GPP. Although it is defined within the LTE standard, many of the characteristics of this standard (handover, channel monitoring or dual connectivity) are reduced

to maintain running costs low. This technology works within the licensed frequency bands assigned to LTE transmissions. There are three methods of work in these frequency bands, also illustrated in Figure 1-3:

- Stand alone: using GSM frequencies, this mode uses the guard bands (10kHz) of the GSM bandwidth (200 kHz).
- Guard-band: Use the unused resource blocks on the LTE guard band.
- In-band deployment: utilizing resource blocks within an LTE carrier. The allocation of resources is not fixed and is not allowed at all frequencies to avoid conflicts with standard LTE operation.

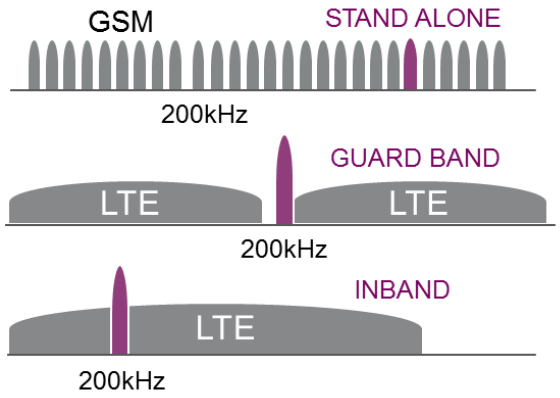

Figure 1-3 Operation modes of NB-IoT

1.3 Low Power Wide Area Networks (LPWANs)

LPWANs are a class of networks characterized by long range and reduced power consumption. These performances are typically obtained by leveraging innovative modulation schemes and by compromising on throughput to achieve an increased communication range. The major industrial representatives of LPWANS are SigFox, Weightless, and LoRaWAN.

1.3.1 SigFox

SigFox is a proprietary technology developed by the homonymous company, founded in 2009. SigFox is an IoT network in which base stations are able to listen to smart objects broadcasting data in the wireless channel, without

needing them to be connected to a peculiar base station. The fact that objects do not need to establish and maintain network connections to transmit data like in cellular networks provides quicker access and inexpensive communication, at the cost of increased interference and the consequential need for a more complex demodulation scheme. The average number of base stations that can be reached by a device is 3 in typical deployments.

Figure 1-4 The SigFox logo

SigFox deploys an Ultra Narrow Band (UNB) modulation in both directions and operates on the publicly available ISM band. This modulation allows long range communication between a client device and a base station. UNB is based on an efficient use of the spectrum that allows greater capacity of the network with a lower consumption, using channel bandwidths smaller than 1 kHz. SigFox uses a custom designed protocol which only works with small messages. This implies relevant energy savings, compared to the transfer of longer messages: maximum payload sizes of 12 bytes in the uplink and 8 bytes in the downlink are allowed, with a protocol overhead of 26 bytes.

This technology was designed primarily for use in the uplink, although it also allows the optional use of a downlink that is used to confirm the packets from the base station to the end devices. The Sigfox network allows adopting different interaction models that differ in the volume of devices and the number of messages that can be transmitted per day [5]:

- Platinum: from 101 to 140 messages + 4 downlinks.
- Gold: from 51 to 100 messages + 2 downlinks.
- Silver: from 3 to 50 messages + 1 downlink.

- One: 1 to 2 messages + without downlink. One of the most significant points is that SigFox gateways pass all data to the proprietary back-end server and the customer portal, so no custom deployments of the technology are possible.

1.3.2 Weightless

Weightless is another communication technology used in the IoT field, developed by Weightless SIG. It is an open standard LPWAN technology that offers long range coverage, long battery life and a low cost of manufacturing devices. It provides encryption and authentication to encode the transmitted information. The technology supports mobility between networks, which automatically routes messages to the correct destination.

Figure 1-5 The Weightless logo

There are three variants of this technology: Weightless-N, Weightless-P and Weightless-W, in order from least to most complex.

1.3.2.1 Weightless-N

Weightless-N, based on the European Telecommunications Standard Institute Low Throughput Networks (ETSI LTN), operates in the free ISM sub-GHz bands (169/433/470/780/868/915/923 MHz) using a DBPSK digital modulation to transmit within narrow frequency bands, and using a frequency hopping algorithm to mitigate interference. Weightless-N is defined by the group itself as a standard to be applied when low cost is the priority.

1.3.2.2 Weightless-P

Weightless-P works in the unlicensed sub-GHz ISM bands with a GMSK modulation using FDMA and TDMA transmission techniques in 12.5 kHz narrow band channels. It performs synchronization at the base station level, making it easier to program the radio channels that are used. It uses the frequency reuse technique, which provides flexibility in channel allocation, thus improving the capacity of the network. It also allows bidirectional transmissions with an uplink and downlink speed of 200 bps and 100 kbps.

1.3.2.3 Weightless-W

Weightless-W is an open standard communication technology that operates in the free spectrum between television bands (400 - 800 MHz). It uses 16-QAM or BPSK modulation with 16 FDMA interspersed channels for America and 24 parallel FDMA channels for the rest of the world, at a rate of 128kbps.

The range of coverage that this technology has is of at least 5km in both urban and open environments, at a transmission speed between 1 kbps and 10 Mbps. These architectures have a network and infrastructure cost greater than the rest of LPWAN technologies [6].

A summary of the main differences between different Weightless versions can be found in table 1:

	Weightless-N	*Weightless-P*	*Weightless-W*
Directionality	One way	Two way	Two way
Band	ISM sub-GHz bands	ISM free bands	400 – 800 MHz
Modulation	DBPSK	GMSK	16-QAM o BPSK
Range	5km+	2km+	5km+
Network cost	Very low	Medium	Medium

Table 2 Comparison of Weightless versions

2 LoRa and LoRaWAN

LoRa/LoRaWAN are one of the main technologies in LPWAN, since they fulfill some necessary requirements for IoT applications, such as bi-directional communications, low complexity for the installation and configuration of sensors and thanks to the freedom it offers users for its deployment.

This type of network has excellent characteristics such as a high energy efficiency, a low energy consumption, integrated security features and a high coverage capacity at the expense of the data rate.

LoRa defines a physical layer technology developed by Cycleo in 2010, a company that was later acquired by Semtech. On the other hand, LoRaWAN is a network specification proposed by the LoRa Alliance in 2015 to offer a MAC layer based on the LoRa modulation.

2.1 LoRa

LoRa (Long Range) is a proprietary physical layer (PHY) design used in the LoRaWAN specification. LoRa is a modulation technique based on the spread spectrum technique, a variation of Chirp Spread Spectrum (CSS). It operates on the ISM bands, specifically on the 433MHz, 868MHz and 915MHz bands, depending on the region in which it is deployed. Cyclic redundancy coding (CRC) and Forward Error Correction schemes are used to ensure correctness of transmitted data. The LoRa modulation makes long range communication possible thanks to an increased sensitivity of the receiver, using the entire bandwidth of the channel to make the transmitted a signal robust against channel noise and path loss.

2.1.1 Spread Spectrum versus Narrowband

The two most important modulation technologies in the area of radio frequency for the robust transmission of information over long distances and

that we have already mentioned above are Ultra-Narrowband and Spread Spectrum. Making a brief comparison between them:

- Ultra-Narrowband is based on a modulation of the signal to be transmitted in a very small range of frequencies. This causes the transmission speed to decrease. SigFox and NB-IOT use this technology.
- Spread Spectrum is based on a widening modulation of the signal, that is transmitted along a very wide band of frequencies. LoRa is based on this technology.

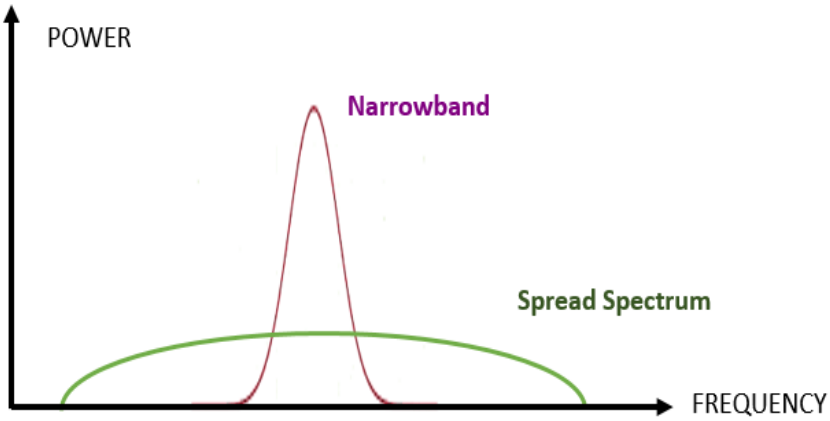

Figure 2-1 Narrowband vs Spread Spectrum

The technology used by LoRa, Spread Spectrum, features some advantages when compared to other long-range modulation technologies:

- Coexistence with narrow band signals: narrowband packets only contribute a small increase in signal noise for a spread spectrum transmission.
- Attenuation of the multipath effect.
- Resistance to a considerable part of the interference.
- Frequency band sharing with other users.

Some disadvantages include that a large bandwidth is needed, and the need of a precise synchronization between transmitter and receiver.

2.1.2 Lora's Chirp Spread Spectrum (CSS) Implementation

CSS is based on usage of a chirp signal (i.e., a sinusoidal signal of linearly variable frequency and fixed duration) to send information occupying a spectrum bigger than strictly necessary. Using a larger bandwidth, it is possible to provide sent symbols a resistance to noise and interference.

One of the main parameters is the Spreading Factor (SF), which represents the number of bits that are coded in a symbol, and is related to the bitrate that is achievable by the transmission. A chirp using a spreading factor value of SF represents 2^{SF} bits using a symbol, and that there are $M = 2^{SF}$ symbols that can be sent by the transmitter. The SF parameter also influences the duration of the symbols, which can be computed using the following expression:

$$T_S = \frac{2^{SF}}{B}. \qquad (2.1)$$

Using a fixed bandwidth, for each increment in 1 of the SF, the symbol duration doubles. This increase makes the message more robust against interference or noise, but on the other hand a longer transmission time increases the possibility of a collision.

The choice of SF affects the sensitivity of the receiver, that is defined by the following expression:

$$S = -174 + 10log_{10}(B) + NF + SNR \qquad (2.2)$$

where the first term is due to thermal noise at the receiver in 1 Hz of bandwidth, NF is the noise figure at the receiver (which is fixed for a given hardware setup), and SNR is the signal to noise ratio required by the underlying modulation scheme [7]. As the bandwidth is set in steps of powers of 2, we can derive from expression ($S = -174 + 10log_{10}(B) + NF + SNR$ (2.2) that increasing the bandwidth decreases the

sensitivity by 3 dB and vice versa. Similarly, increasing the spreading factor increases the sensitivity by 3 dB.

Starting from the expression CSS is based on usage of a chirp signal (i.e., a sinusoidal signal of linearly variable frequency and fixed duration) to send information occupying a spectrum bigger than strictly necessary. Using a larger bandwidth, it is possible to provide sent symbols a resistance to noise and interference.

One of the main parameters is the Spreading Factor (SF), which represents the number of bits that are coded in a symbol, and is related to the bitrate that is achievable by the transmission. A chirp using a spreading factor value of SF represents 2^{SF} bits using a symbol, and that there are $M = 2^{SF}$ symbols that can be sent by the transmitter. The SF parameter also influences the duration of the symbols, which can be computed using the following expression:

$$T_s = \frac{2^{SF}}{B}. \quad (2.1$$

we can also compute the bitrate for a certain pair of SF and B in the following way:

$$R_b = \frac{SF}{T_s}$$

The bitrates for a range of spreading factor and bandwidths can be found in the following table:

SF	135kHz	250kHz	500kHz
7	6835	13671	27343
8	3906	7812	15625
9	2197	4396	8793
10	1220	2441	4882
11	671	1342	2685
12	366	732	1464

Table 3 Bitrates comparison with SF and bandwidths

2.1.3 LoRa Physical Layer Packets

Figure 2-2 represents a spectrogram where the horizontal axis represents time and the vertical axis the frequency. At the PHY layer, a LoRa message is based in the chirp signal sweeping the frequency band.

Figure 2-2 Spectrum representation of a LoRa signal

The signal is composed, first of all, of a preamble (minimum length of 4.25 chirps) where several chirps sweep the frequency band. Secondly, the data is modulated in a signal based on instantaneous changes in the frequency of the chirp. In the decoding process, the signal is first “de-chirped”, and then a Fast Fourier Transform (FFT) of the signal is taken, with a number of bins equal to the number of symbols M that corresponds to the used spreading factor. The signal that is obtained can be interpreted as a signal that was modulated using Multiple Frequency Shift Keying (MFSK), as shown in Figure 2.2.

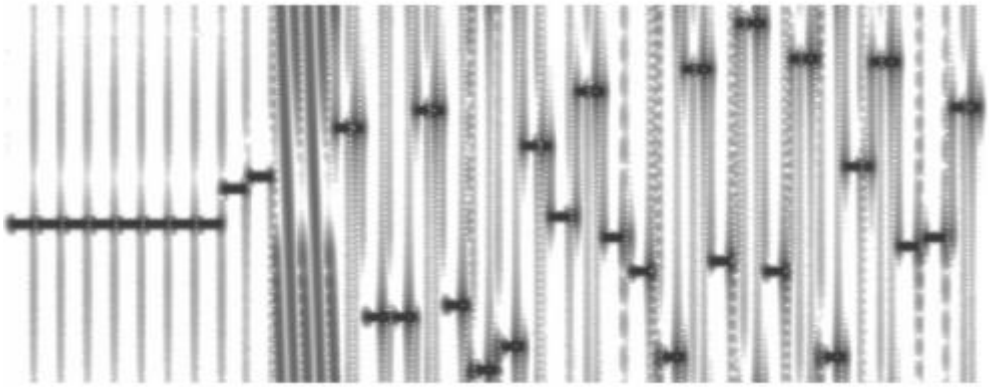

Figure 2-3 De-chirped version of the LoRa signal

Moreover, LoRa also specifies different encoding operations that can be applied before modulation and transmission:

- Data whitening is used to avoid the transmission of long equal bit runs and to distribute information across the used bandwidth.

- Forward Error Correction (FEC) is implemented as a Hamming Code, in order to correct errors. Available code rates for a LoRa packet are C ∈ {4/5; 4/6; 4/7; 4/8}.

- Interleaving scrambles the output of FEC, in order to make the code more resistant to errors.

- Grey mapping is used to map a block of SF bits into one of the M symbols in the constellation.

The parameters described above influence the on-air time of a packet. On the one hand, $t_{preamble}$ is the time it takes to send the preamble and is obtained as:

$$t_{preamble} = \left(n_{preamble} + 4.25 \right) \cdot t_s$$

where $n_{preamble}$ is a configurable parameter that affects the number of symbols in the preamble.

$t_{payload}$, then, is the time that is necessary to transmit the data, and can be obtained with the following expression:

$$t_{payload} = n_{payload} \, . t_s$$

$$n_{payload} = 8 + max\left(\left[\frac{8PL - 4SF + 44 - 20H}{4(SF - 2DE)} \right] (CR + 4) \, , 0 \right)$$

where $n_{payload}$ is a parameter that depends on the number of payload bytes PL, H depends on whether the PHY header is disabled (0) or enabled (1), DE depends on whether the low data rate optimization is disabled (0) or enabled (1) and, lastly, CR that is the number of added parity bits.

Finally, is possible to calculate on-air time as show the next expression:

$$t_{packet} = t_{preamble} + t_{payload}$$

As we mentioned earlier, LoRa bases its physical layer on CSS, which integrates Forward Error Correction (FED). Thanks to the use of broadband, interferences are reduced. A LoRa receiver can decode transmissions 19.5 dB below the noise floor. Thus, very long communication distances can be bridged [8].

A LoRa radio has four configuration parameters that determine the consumed energy, the transmission range and the noise resilience. They are the following:

- Carrier Frequency: this is the central frequency used in the transmission.
- Spreading Factor: a higher spreading factor increases the Signal to Noise Ratio (SNR), the sensitivity and also the time that the package is in the air. Spreading factor can be adopted from 6 to 12, with the former having the highest transmission rate. It is possible to have a network with different SFs, and this contributes to generating fewer collisions, since different SFs are quasi-orthogonal, i.e., overlapping packets using different SFs can be demodulated correctly.
- Bandwidth: the range of frequencies in the transmission band. Greater bandwidth gives a higher data rate, but at the same time a lower sensitivity.
- Coding Rate: Coding Rate (CR) is the FEC rate used by the LoRa modem. A higher CR offers more protection but increases time on air.

Radios using different CF, SF, BW and CR, can still communicate with each other.

2.1.4 Interference

For the simulation of a LoRaWAN network, the interference that comes from other LoRa transmissions can be modeled by considering the partial orthogonality property of different SFs, and deciding when a packet survives interference or not based on the overlap and powers of the colliding packets.

The following Signal to Interference Ratio (SIR) threshold matrix, which depends on the spreading factor, can be used to model interference [9]:

$$\mathbf{T} = \begin{bmatrix} 6 & -16 & -18 & -19 & -19 & -20 \\ -24 & 6 & -20 & -22 & -22 & -22 \\ -27 & -27 & 6 & -23 & -25 & -25 \\ -30 & -30 & -30 & 6 & -26 & -28 \\ -33 & -33 & -33 & -33 & 6 & -29 \\ -36 & -36 & -36 & -36 & -36 & 6 \end{bmatrix}$$

Element $T_{i,j}$ in the matrix above is the SIR margin (in dB units) that a packet sent with $SF = i$ must have in order to be correctly decoded if the interfering packet has $SF = j$. In other words, the matrix expresses the fact that two devices using different spreading factors can transmit their data simultaneously, provided some condition on the SIR is respected. In presence of multiple interfering packets, it is necessary to satisfy the margin conditions with all the interfering packets, summing the received power values for each SF [7].

Focusing on the Single Input Single Output (SISO) case, the general definition of Signal to Interference plus Noise Ratio (SINR) is:

$$SINR = \frac{P_{rc,0}}{\sigma_w^2 + \sum_{l=1}^{N_{int}} P_{rc,l}} \qquad 2.1$$

Where the variables are:

- $P_{rc,0}$: power of the packet
- N_{int} : number of interfering packets
- $P_{rc,l}$: power of the l-th interfering packet

If we focus from the general case to the case of an end device which uses $SF = i$ and a set I_j of interferers using $SF = j$, it is possible to see which packets are able to be correctly decoded with $SF = i$ for every j as shown in the following expressions:

$$SINR = \frac{P_{rc,0}}{\sigma_w^2 + \sum_{l \in I_j} P_{rc,l}} \qquad 2.2$$

$$SINR_{i,j}^{dB} > T_{i,j} \qquad 2.3$$

The values of the matrix T are calculated on the assumption that the packages are completely overlapping. But this is not always the case, so the interfering power value for the computation of the SINR needs to be 'equalized' based on the amount of overlap between the packets.

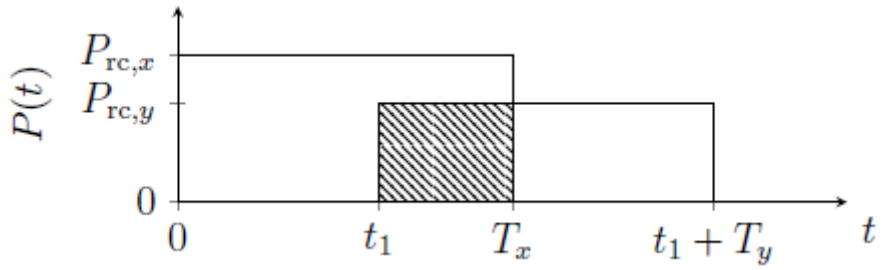

Figure 2-4 Power equalization of colliding packets. The highlighted energy is spread on the duration of the packet

Considering the situation shown in Figure 2-4, the first packet, which uses $SF = x$, is received in t_o and whose time of transmission is T_x, it is possible to calculate the energy of the packet as $E_x = P_{rc,x} \cdot T_x$. On the other hand, the interfering energy of a second packet, which uses $SF = y$, is received in t_1 and whose time of transmission is T_y, can be computed this way: $E_y^{interf} = P_{rc,x} \cdot (T_x - t_1)$. Therefore, the equalized interfering power is:

$$P_{rc,y}^{interf} = \frac{E_{rc,y}^{interf}}{T_x} = \frac{P_{rc,x} \cdot (T_x - t_1)}{T_x} = P_{rc,y}\left(1 - \frac{t_1}{T_x}\right)$$

Using the computed interference power and the period of time during which both packets are overlapped and the general equation $SINR = \frac{P_{rc,0}}{\sigma_w^2 + \sum_{l=1}^{N_{int}} P_{rc,l}}$ 2.1, the following is obtained :

$$P_{rc,y}^{interf} = \frac{P_{rc,x} \cdot t_{ol}}{T_x}.$$

This assumption to pass from completely overlapping to partially overlapping packets is justified by the fact the underlying channel code employed by the

modulation makes use of an interleaver which allows the channel code to eventually correct the errors caused by the interference.

The T matrix allows us to obtain some conclusions about the interference between LoRa packets:

- Two packets sent with the same SF and a similar receive power can still be decoded correctly. This fact depends on the time that both are overlapping, because it must be sufficiently small for the SINR of both to be above 6dB.
- Not all SF values are as resistant as others against interferences. Generally speaking, it can be seen that an increase of one unit in SF, provides 3db of resistance against interference.

2.1.5 Spreading Factor Orthogonality

In order to improving the potential of the network and increasing the capacity of the network in spite of collisions between packets, the LoRa modulation features six orthogonal Spreading Factors (SF) resulting in different data rates. Signals with different spreading factors can be distinguished and received simultaneously, even if they are transmitted at the same time on the same central frequency and bandwidth, without degrading the communication performance. This happens because different SF are orthogonal. In other words, this fact allows a receiver to correctly detect a packet using spreading factor even if it is overlapping in time with another transmission employing spreading factor j, as long as i and the received packet's Signal to Interference plus Noise Ratio (SINR) is above a certain threshold (also called isolation) that depends on both i and j[7].

2.2 LoRaWAN

LoRaWAN is the specification defining the MAC protocol and network architecture for networks of devices employing the LoRa modulation. The specification is proposed by the LoRa Alliance, focusing on the basic requirements of IoT, such as secure two-way communication, mobility and location services [10].

2.2.1 LoRa Network Architecture

LoRaWAN networks extend into a star-of-stars topology where gateways relay messages between nodes and a Network Server (NS), located in the back-end. The gateways are connected to the server through a standard IP connection, while the nodes and the gateways establish their connection through a direct link using LoRa at the physical level.

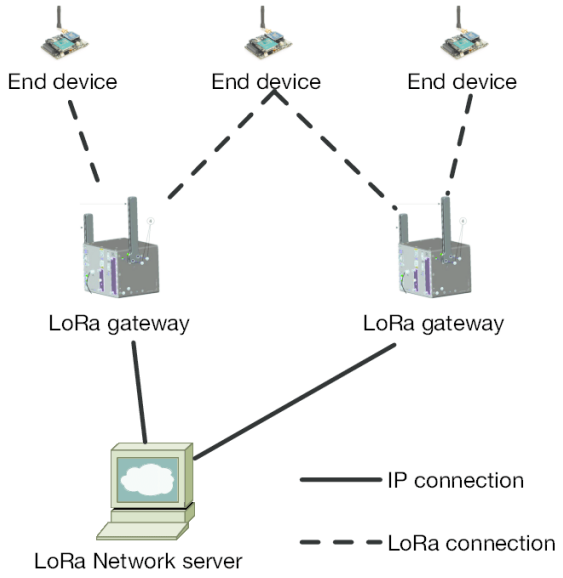

Figure 2-5 LoRaWAN network architecture

2.2.2 Components of a LoRaWAN network

There are several components that are necessary to form a LoRaWAN network.

- End-devices (EDs): these are the devices that communicate with the gateways through the LoRa physical layer technology. Typically, these are sensor or actuator devices.

- Gateways (GWs) allow the interconnection between networks with different protocols and architectures. Their purpose is to translate the information of the protocol used in a network to the protocol used in the destination network. Gateways are bidirectional relays, or protocol converters. LoRaWAN frames are sent to the Network Server over a

backhaul interface with a higher throughput, and the NS is responsible for decoding the packets sent by end-devices and generating the packets that should be sent back to the nodes, like acknowledgements or commands. Within a LoRaWAN deployment there may be several gateways.

- The Network Server (NS) decodes the packets received by the gateway that have been sent by the end-devices. It is responsible for sending back acknowledgements when required by the end-devices. The end devices are not associated with any particular gateway, so the gateways only act as a relay between the end device and the network server to which it is associated, the latter being able to choose the best option among all gateways for transmitting a response if one is necessary.

The LoRaWAN V1.0 specification specifies 3 classes of end devices, depending on how the downlink is managed. All LoRaWAN devices implement at least Class A functionality, which is mandatory. In addition to this class, they can implement Class B or Class C options.

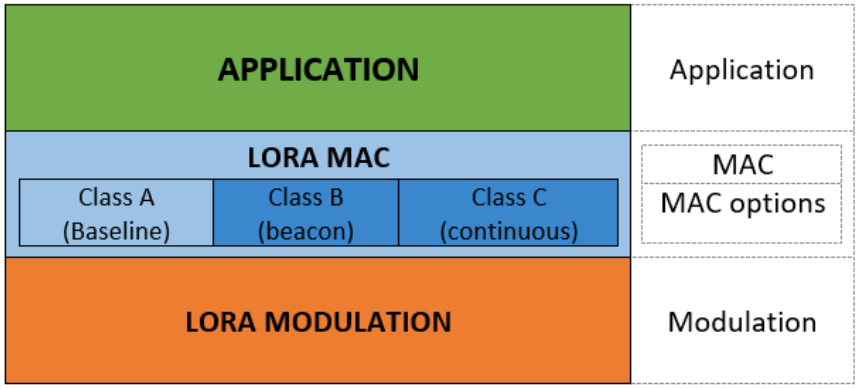

Figure 2-6 LoRaWAN protocol stack [10]

- Class A: Bi-directional end-devices. As shown in Figure 2-7, Class A devices can start communication in the upstream channel. Only when a broadcast has been sent is the downlink channel enabled, by the opening of receive windows, to receive acknowledgments (ACKs) or

application data if required. The second window is not necessary if a packet is received in the first window. Any of these two downlink windows can be used to transfer data from the base station to the respective end devices, and they can be configured to use a different frequency channel and SF from the one used in the uplink, to avoid interference.

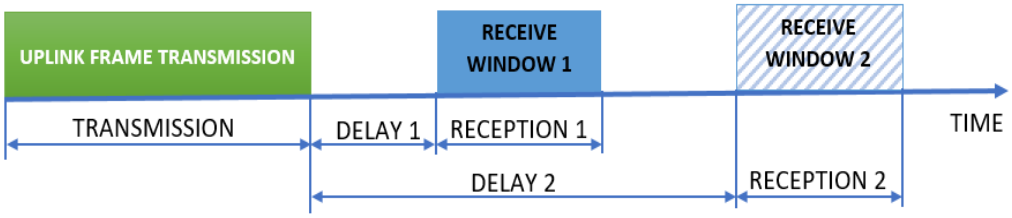

Figure 2-7 LoRaWAN class A transmission phases [11]

- Class B: Bi-directional end-devices with scheduled receive slots. With these devices, in addition to the two reception windows enabled with devices A, special reception windows can be periodically enabled at scheduled times. This is achieved through the diffusion of synchronized beacons by the base stations that give a reference in time and provide a periodic synchronization for the end devices. The network can send packets to type B devices in any of the reception slots enabled in the downlink.

- Class C: Bi-directional with maximal receive slots. The end devices remain continuously in a listening state. This is the class that consumes the most energy.

2.2.3 Packets structure

The standard also defines the format of PHY and MAC layer packets. As shown in Figure 2-8, at the PHY layer the format of the packet is composed of a preamble, a header, a payload and two CRC sections, to protect header and payload respectively. The PHY layer payload contains information about the version of the standard that is being used about and the type of message.

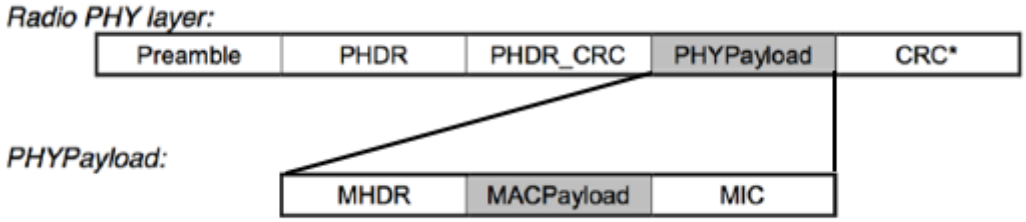

Figure 2-8 LoRaWAN Packet structure

The MAC payload contains three parts, as shown in Figure 2-9:

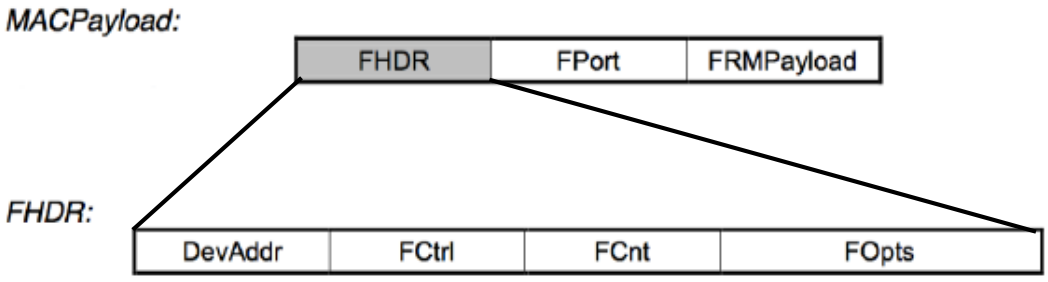

Figure 2-9 MAC layer packet structure

- Frame payload: contains data coming from the Application Layer.
- Frame port: used for the identification of what application the packet is meant for. Some port numbers are reserved.
- Frame header: contains different fields about different aspects of a LoRa network. The first 4 bytes are a devices address used to identify the device inside the network. Then, one byte, called Frame Control Field, contains information about acknowledgement management. The last part are 2 bits that are used to implement the Adaptive Data Rate (ADR) feature. Using it, the NS is able to change the spreading factor employed in the transmission depending on the needs of the network. It should be noted that the ADR algorithm is not standardized, so it can be implemented in different ways. If this feature is enabled, the devices know that they should follow the instructions on the MAC commands contained in the FOpts field. These MAC commands, which are exchanged between the end-devices and the NS, can be used to gather information about link quality, handle ADR requests and replies, impose

limitations to a device's transmit time, configure parameters of the receive windows and configure the available radio channels.

3 SIMULATION OF LORAWAN NETWORKS

Using simulators enables studies that are more difficult or not possible to perform with real systems, to observe system behavior in a highly controlled, reproducible environment, and to gain insights about how networks work. For research focused on networks and communication protocols, the choice of a good simulator takes on a vital importance, where high reliability in the results needs to be balanced with possibly high computational costs.

Some examples of network simulators include:

- OPNET (Optimized Network Engineering Tools): extensive simulation software with a wide variety of possibilities to simulate entire heterogeneous networks with various protocols.
- NETSIM (Network Based Environment for Modelling and Simulation): an application that simulates Cisco Systems networking hardware and software, and is designed to aid the user in learning the Cisco IOS command structure.
- OMNET++ (Objective Modular Network Testbed in C++): an extensible, modular C++ simulation library and framework.
- JSIM (Java-based simulation): Java-based simulation system for building quantitative numeric models and analyzing them with respect to experimental reference data.

For the realization of this project the ns-3 simulator is used, due to it is open source release model, its high flexibility and due to the availability of a lorawan model.

3.1 NS-3

The ns-3 simulator is a network simulator targeted primarily for research and educational use. The ns-3 project, started in 2006, is an open-source project developing ns-3. Simulators of the "Network Simulator" family (ns-1, ns-2 and

ns-3) are a series of simulators for networks based on discrete events: systems are modeled as a series of discrete events, which are sequentially executed. Network Simulator 3 (ns-3) is a simulator developed in C ++ as a successor to ns-2 and was built on a similar base as his predecessor, however there is no compatibility between modules developed for the two simulators.

ns-3 allows the definition of various types of both cabled and wireless network topologies, such as Ethernet, Wi-Fi and other technologies, allowing the user to configure all individual components of the network; on these topologies, it is possible to program the exchange of data using real protocols that have been implemented in ns-3.

3.1.1 Conceptual Overview

3.1.1.1 Organization

The simulator core and models in ns-3 are implemented in C++. ns-3 is built as a library, which may be statically or dynamically linked to a C++ main program that defines the simulation topology and starts the simulator. ns-3 also exports nearly all of its API to Python, allowing Python programs to import an “ns3” module in much the same way as the ns-3 library is linked by executables in C++ [12].

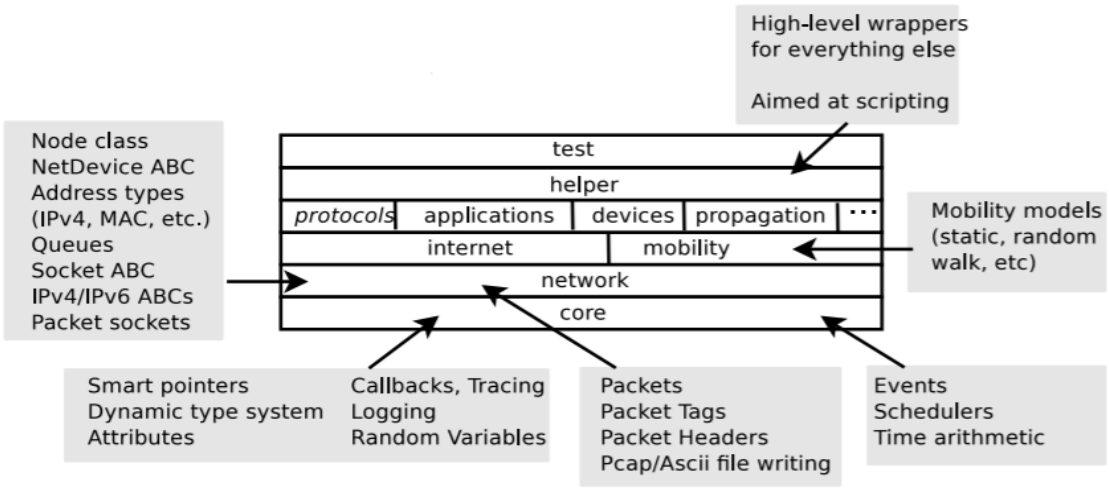

Figure 3-1 ns-3 software organization

The source code for ns-3 is mostly organized in the *src directory* and is described in Figure 3-1. In general, modules only have dependencies on modules beneath them in the figure. The core of the simulator is implemented in src/core, and includes the simulator engine that executes events and components that are common across all protocol hardware. The network model, implemented in src/network, is where packet objects are defined.

3.1.1.2 Main concepts

3.1.1.2.1 Node

The basic computing device abstraction is called the node, and its abstraction is represented in C++ by the class Node. The Node class provides methods for managing the representations of computing devices in simulations. Nodes can be characterized by information about their mobility model (i.e., their position and velocity) and by the availability of certain protocol stacks for communication with other devices.

3.1.1.2.2 Application

In ns-3 applications are the basic abstraction for a user program that generate some activity (i.e., traffic) in the network. This abstraction is represented in C++ by the generic class *Application*, which can be specialized to create new applications and generate a precise traffic model [13].

3.1.1.2.3 Channel

Nodes can be connected to objects representing a communication channel. This the basic communication subnetwork abstraction is modeled in C++ by the Channel class, which provides methods for connecting nodes and to deliver packets to all connected devices, computing propagation losses and delays according to a specific model and the positions of the nodes.

3.1.1.2.4 Net device

A network device is “installed” in a Node in order to enable the Node to communicate with other Nodes in the simulation via Channels. Just as in a real computer, a Node may be connected to more than one Channel via

multiple NetDevice objects. The NetDevice class provides methods for managing connections to Node and Channel objects, and to send and receive packets through the channel.

3.1.1.2.5 Topology helpers

Since connecting large numbers of NetDevices, Nodes and Channels, assigning IP addresses and so on are common repetitive tasks in ns-3, topology helpers exist to automate the configuration process and perform some actions on a wide array of objects.

3.1.1.3 Modeling network elements

ns-3 has models for all the various network elements that comprise a network. In particular there are models for [14]:

1. Networks nodes, which represent end-systems such as network routers, hubs and switches.
2. Network devices which represent the physical device that connects a node to communications channel.
3. Communications channels, which represent the medium used to send the information between network devices.
4. Communications protocols, following the model of the implementation of protocol descriptions. These protocol objects typicall are organized into a protocol stack where each layer in the stack performs some specifically and limited function on network packets.
5. Protocol headers which are subsets of the data found in network packets.

Network packets are the fundamental unit of information exchange in computer networks. Nearly always a network packet contains one or more protocol headers, describing the information needed by the protocol implementation at the end points. Further, the packets typically contain a payload which represents the actual data being sent between end systems.

In addition to the models for the network elements mentioned above, ns-3 has a number of helper objects that assist in the execution and analysis of the simulation but are not directly modeled in the simulation. There are:

1. Random variables, which can be created and sampled to add the necessary randomness in the simulation.

2. Trace objects, to obtain performance data from the simulation and extract network performance analyses. These traces can be created in various formats such as PCAP, which can be analyzed with tools like WireShark. In addition, there are other formats such as the simple format, where the traces about the flow of packets in the simulated network are more legible.

3. Helper objects are designed to assist with and hide some of the details for various actions needed to create and execute an ns-3 simulation.

4. Attributes allow to configure most of the network element models. Upon creation, network element models have default values that can be changed by specifying their new values in the command line when running the ns-3 simulation or calling specific API functions provided by the objects.

3.1.1.4 Main steps of a simulation

ns-3 simulation programs follow these four main steps:

1. Creation of a topology: it is necessary to instantiate objects for the main elements of the network like those that have been mentioned before, including nodes, devices, channels and networks protocols.

2. Models: the protocol stack is installed on the created objects. For this reason, helper classes are used, which assist in the installation of several objects implementing the needed layers of the ISO / OSI stack. This way, it is possible to create simulation models that are able to send and receive information and create, accept and process packets.

3. Configuration: these models are configured at the beginning with a default values which can be adapted to the requirements of the simulated network and environment. New links between the nodes can be created.

4. Execution: the simulator executes the events and functions specified by the object system that was created in the previous steps.

5. Performance analysis: when the simulation ends, generated data can be collected for analysis and visualization of the results.

3.1.1.5 Pseudo Random Number Generator

ns-3 also provides a built-in Pseudo Random Number Generator (PRNG) like described in [15]. It is called Pseudo because it not truly random, since the number sequence is determined by an initial value (called the PRNG seed). The generator creates $1.8 \cdot 10^{19}$ independent streams of random numbers, each one having $2.3 \cdot 10^{15}$ substreams. Objects that access each stream of the underlying random number generator thus yield an independent output one from each other.

As an example, this function is implemented in ns-3 in the *UnifomRandomVariable* Class. This class is able to create objects that return random numbers between two numbers of type double, between the values of -1.79769e+308 and 1.79769e+308. This class was used during this work to generate random delays and positions of the end devices.

3.1.2 Models for LoRa Networks

This thesis leverages the LoRaWAN module provided by [16]. The module is composed of different classes that describe the behavior of Lora and devices and gateways. Both EDs and GWs must be defined at various levels, from the PHY to the Application level. For this reason, it is necessary to simulate the whole protocol stack.

Also, it is necessary to recreate a real environment to get results that are as close to those of a real network as possible. A set of other classes represent the problems that a communication between devices could have, such as interferences, duty cycle limitations, presence of obstacles in the wireless channel, etc.

3.1.2.1 PeriodicSender

The application layer class PeriodicSender consists in a packet generator, which creates zero-filled packets of a random payload size (between 20 and 200 bytes) and sends them periodically. This class exposes an attribute allowing to tune the delay between transmitted packets. The transmission period is stored in a private attribute called m_interval.

3.1.2.2 AlarmSender

The application layer class AlarmSender is very similar to the previous application layer class. It creates a single packet with the same characteristics. AlarmSender has been created as part of the original work in this thesis to model a different behavior of multiple devices that will synchronously send a single packet after a certain event happens.

3.1.2.3 LoraMac

The LoraMac class models the MAC layer of a LoraWAN devices. This class is the class that is responsible to forward messages coming from the application layer to the PHY layer. Also, a DutyCycleHelper class was created to model duty cycle restrictions and don't allow sending of packets too often. Two additional subclasses, EndDevicesLoraMac and GatewayLoraMac, implement behaviors of the ED and the GW, respectively.

EndDeviceLoraMac is the object where the behaviors associated to the class of the ED are implemented. In this class it is possible to modify the parameters of transmission at the PHY layer, and also to configure the receive windows, that must be opened at precise time frames to receive packets from the gateway. Class A behavior is already implemented in the class, and this is the

portion of the code that should be modified to also support Class B and C devices.

The GatewayLoraMac class differs from EndDeviceLoraMac in that it implements a simpler, forward-only MAC layer. Both classes must be able to understand and act according to the MAC commands that they exchange.

3.1.2.4 LoraPhy

The LoraPhy class models the physical layer of a LoRa device. This class delivers packets to other devices through the LoraChannel class. Furthermore, it decides whether a packet obtained from the channel is correctly received, based on its power and the interference the device is experiencing.

3.1.3 Code

This section describes the code that was used to perform the simulation of a lorawan network.

3.1.3.1 Complete Network Performances

Command line arguments

ns-3 allows us to override default attributes via command line arguments, simply by declaring a command line parser object in the main program. In this case, the chosen attributes are the following:

- Number of periodic end devices to include in the simulation
- Number of alarm end devices to include in the simulation
- The radius of the area to simulate
- The duration of the simulation
- The period in seconds to be used by periodically transmitting applications
- The number of application periods to simulate
- The number of periods we consider as a transient
- The maximum number of allowed transmissions

- Whether to enable mixed application periods, or to use the same period for all devices
- Whether or not to print a file containing the ED's positions
- Percent of devices that will require an acknowledgement

This structure allows easy execution of multiple simulations with different parameters.

Create the channel

To model the channel the LogDistancePropagationLossModel class is used. This model is available by default in ns-3, and computes the reception power using the following expression:

$$L = L_0 + 10n \log_{10}\left(\frac{d}{d_0}\right)$$

where n is the path loss distance exponent, d_0 is reference distance (m), L_0 is path loss at reference distance (dB), d is distance (m) between the sender and the receiver and L is the path loss (dB). In our case, we use the following values: $n = 3.76$, $d_0 = 1$ and $L_0 = 8.1$, while d is computed for each transmission based on the position of transmitting and receiving nodes.

Moreover, it is necessary to calculate the propagation delay experienced between a specified source and destination. In order to do so, the PropagationDelayModel class is used.

Using loss and delay models as parameters, the channel is created.

Create the helpers

Three helpers are created in this part of the script. These helpers are LoraHelper, LoraPhyHelper and LoraMacHelper, and were designed to install and configure the Lora stack on a set of nodes. Moreover, these helpers create and deploy LoraNetDevice, LoraMac and LoraPhy objects, automatically configuring the parameters that are needed for communication between devices, connecting the physical layers to the specified LoraChannel.

The LoraPhyHelper needs to be configured to specify the LoraChannel to connect the PHYs to, the kind of PHY it will create (for a gateway or a device). Then, a LoraPhy is created and connected to a NetDevice on a node. It is also necessary to configure, in the case of a gateway, the frequencies where the PHY will listen (default EU channels) and the allocation of the reception paths.

In the case of the LoraMacHelper, first the operational region (in our case EU) is selected, and the corresponding default channels are created. In the case in which an ED is being configured, an address is created to uniquely indentify the device in the network. Then the LoraMac instance is created and connected to a NetDevice where some parameters specific to the EU region are configured. These include the transmission power in dBm, the DataRate with which the gateway will respond to the device, the definition of the Spreading Factor to Data Rate correspondence.

The LoraHelper class simply performs the actions above, using the LoraPhyHelper and LoraMacHelper instances, to install the NetDevice over all the nodes in the provided Node Container.

Create end devices

End devices are created, and a MobilityModel is installed in each node. This object represents the node's physical position and how it changes over time.

Generate the address of devices and connect them to the channel

Using the LoraDevicesAddressGenerator class, we obtain a sequential address for each device. We use this address to set up the MAC of the devices using the LoraMacHelper and, more specifically, configure it via the SetAddressGenerator function.

Connect trace sources

In order to connect the trace sources, the script goes over all the devices that have been created and obtains their LoraPhy objects. The trace sources of these objects are then connected to functions that are defined in the script and that will be called when certain events happen in the simulation. This allows the simulation script to compute the relevant metrics.

Create and configure gateways

A gateway is created and positioned at the center of the simulated space. A ListPositionAllocator is used to specify the (0, 0) coordinates.

Moreover, it is also necessary to create a NetDevice for each gateway. This is done using LoraPhyHelper and LoraMacHelper, indicating that these devices are gateways and not end devices. These helper classes configure the underlying protocol stack model so that the GatewayLoraPhy class can manage its 8 receive paths. After reception of a packet is completed, this class decides whether the packet was lost because interference, because no more demodulators were available or because the received packet arrived under the required sensitivity. Depending on the state, different functions in the simulation script are called through the corresponding trace sources.

Create network server

Using the NetworkServerHelper and the SetGateways function it is possible to configure which gateways will be connected to the NS. Using the SetEndDevices function the NS is informed of the set of EDs it will need to manage. This helper also installs the Forwarder application on the gateways.

Set up the device's spreading factor

To set up the spreading factors used by the devices the LoraMacHelper object's SetSpreadingFactorUp function is used. It takes as arguments a node container of the end devices, the gateways and a pointer to the channel.

3.1 Procedure: Spreading factor assignment

---

1. Set S as a vector containing the sensitivities to each SF
2. For each ED in the network
3. For each GW in the network
4. Compute the received power for the transmission from ED to GW
5. Select the GW that received the strongest signal, having power P
6. Configure the ED to use the lowest SF such that P > S
7. Return

In real deployments, spreading factors would be configured dynamically throughout the network lifetime via MAC commands. However, in this simulation we will use fixed spreading factor values. Starting from this fact, the procedure to assignment SF to each ED, as show the Algorithm 3.1, is implemented in this way: firstly, the power level received from ED to GW is computed. Then, it is verified which one is the GW that receives the highest power from the chosen ED, and the SF is set based on that value. SFs are assigned according to the gateway sensibility as shown in Table 4. This implies that the lowest SF that allows reception is used, because an increase of the SF also increases the Time on Air (ToA), and a higher ToA implies also a higher probability of collisions.

SF	Sensibility (dBm)
7	-127.0
8	-129.5
9	-132.0
10	-134.5
11	-137.0
12	-139.5

Table 4 Sensitivity to different SFs

Install applications on the end devices

Applications are finally installed using the PeriodicSenderHelper object.

3.1.3.2 Complete Network Performances Alarm

Starting from the previous code, a new type of end device, representing an alarm, has been added. Alarm devices will be used to simulate a different situation where some devices that are very close one with another try to simultaneously send packets.

4 RESULTS

In this section we will analyze the results obtained using the lorawan module in ns-3. First of all, the network is evaluated with some probabilities of success and loss of a packet in a LoRaWAN network in the presence of uplink only devices. Then, a network is set up with an acknowledged traffic in order to see how this traffic affects the network performance, in comparison with the unreliable traffic network. Lastly, alarm devices will be added to the downlink network scenario, in order to evaluate the probability of packet success and delay performance. Trying to achieve a better performance, some policies are finally evaluated to mitigate interference between simultaneously transmitting devices.

4.1 OVERVIEW OF CODE AND PLOTTING

With the code explained before, we have the following situation: a single GW is allocated in the center of a circle, whose radius is 7500 m. This radius has been chosen because it is the maximum distance that a device can be at and still communicate with the GW if we consider the propagation loss using the maximum SF=12. Inside of this radius and collocated in a random way are the end devices. The gateway was configured only in uplink mode, i.e., it didn't forward any traffic to or from the Network Server. The period, which in this simulation is 600 seconds, is used by periodically transmitting applications. The initial situation is as shown in Figure 4-1 Distribution of devices and DRFigure 4-1:

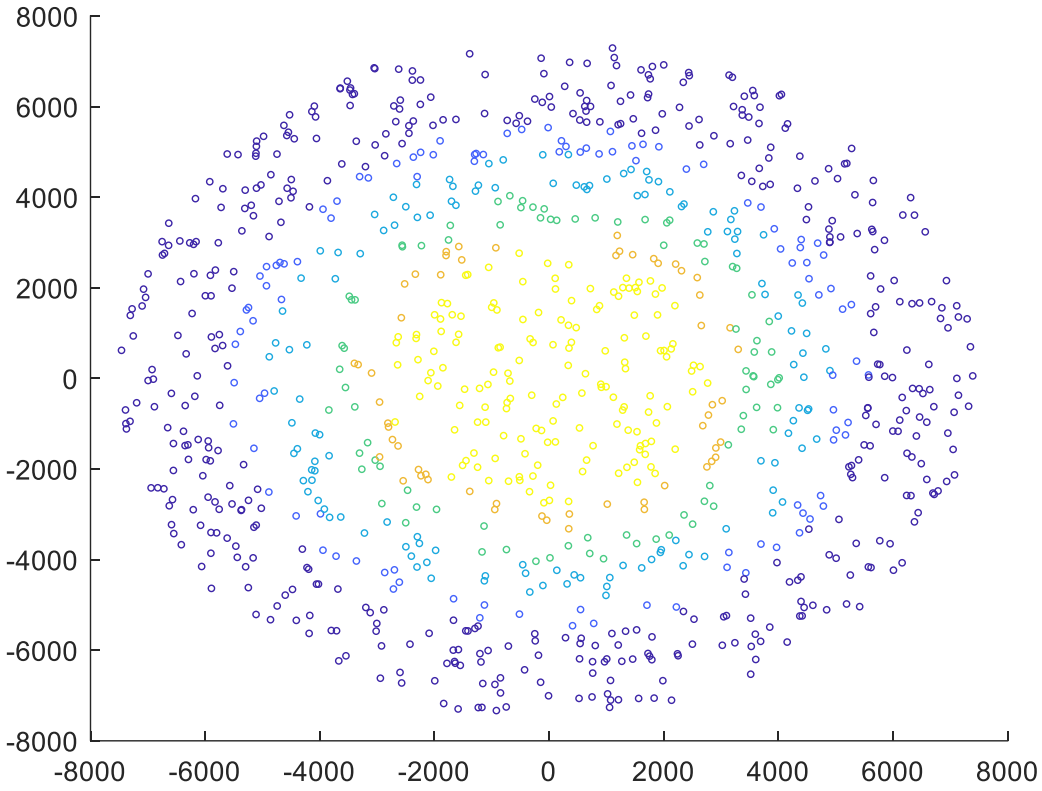

Figure 4-1 Distribution of devices and DR

The colors of the Figure 4-1 represent the different data rates, where devices in the center have the highest data rate. The figure also shows how the choice of the Data Rate (DR) depends on the distance of the end devices to the gateway, as it has been explained in the previous section. Table 5 shows the correlation between SF and DR.

SF	DR
7	5
8	4
9	3
10	2
11	1
12	0

Table 5 Correlation between SF and DR

In this simulation, three statistics were evaluated:

- *Probability of success (P_{succ}):* is the probability that a sent packet is correctly received. This probability is approximated as the ratio between the number of successfully received packets and the total number of sent packets during a simulation. As shown in Figure 4-2, when the network has a low number of devices, the network works with a satisfying probability of success (more than 95%). As the number of devices increases, and with it the traffic, the successful packet probability decreases. Using some threshold for acceptable throughput which depends on the requirements of the network, we could calculate the maximum of devices that could be used in the network.
- *Probability of no more receivers (Pnmr)*: is the probability that a sent packet was not received because no more receive paths were available at the gateway.
- *Probability of interference* (P_{int}): is the probability that a sent packet finds a free reception path but is corrupted by interference. The probability of this case, as can be seen in Figure 4-2 and can be expected, increases with the increase of the devices. One of the reasons this happens is because, although the devices are using different SFs, there are some devices that are transmitting with the SF. Another reason can be the fact that if there are more devices this implies that there are more devices using SF=12, whose transmission time is higher. In turn, this increases the probability of a collision.

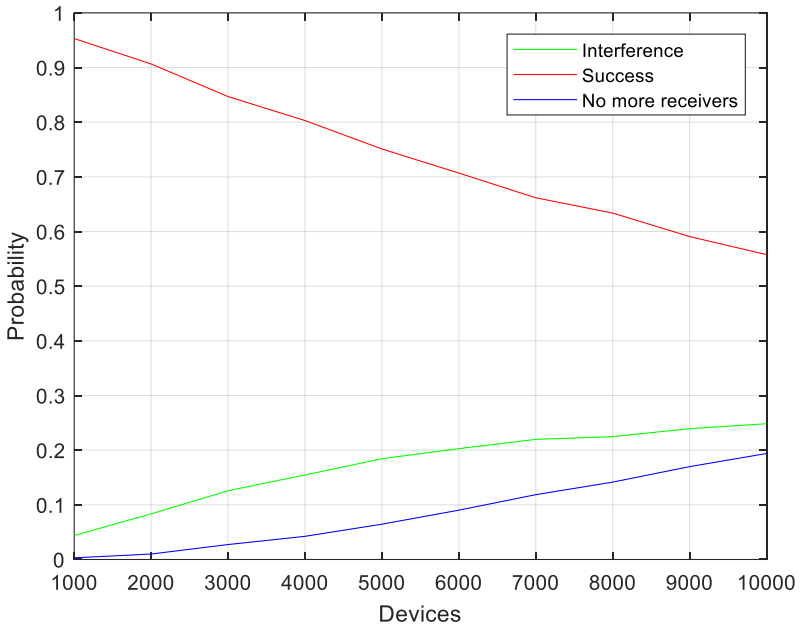

Figure 4-2 Probabilities of success, interference, NMR

4.2 PERCENTAGE OF ACKNOWLEDGED FRAMES

With the last results in mind, this simulation aims at estimating the effect on the uplink packet performance of acknowledged traffic.

4.2.1 Parameters

In this case, the simulation is performed with a variable number of periodic devices, in order to analyze how the performances vary depending on the percentage of devices using confirmed transmission and on the amount of traffic. For these simulations, a fixed amount of periodic traffic was used (with a period of 20 minutes).

4.2.2 Surface plot

Figure 4-3 represents how the probability of success varies with respect to the percentage of packets that require acknowledged frames and the number of periodic devices. As shown in the plot, performance can drastically vary. On the one hand, when the network has few devices, up to 400, the probability of success with any percentage of acknowledged frames is over 80%. We can see that with no confirmed transmission (percentage of ACK of 0%), the curve

that presents the probability of success diminishes around 20% when going from zero devices to two thousand. In contrast, as the percentage of ACK rises, this curve is more abrupt and the probability of success diminishes of around 50%. In other words, the performance with acknowledged frames degrades much more quickly when traffic is heavy.

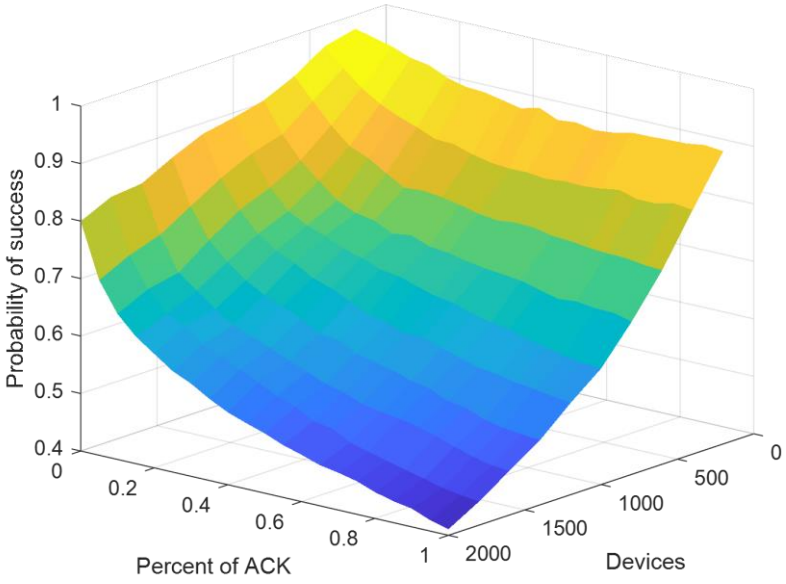

Figure 4-3 Performances acknowledged frames

4.2.3 Contour plot

In the Figure 4-4 we presented the same plot as in the previous case, but instead, this graph allows us to define operating regions based on performance requirements.

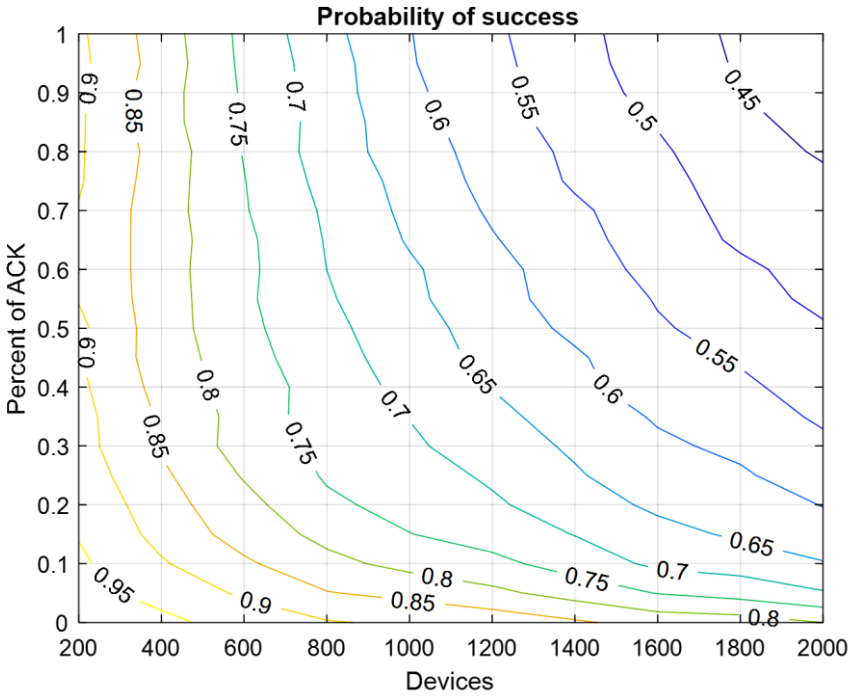

Figure 4-4 Probability of success vs Devices and percentage of ACK

Following the lines which represent the probability of success and selecting which probability is the minimum accepted probability in the network, we can know how many devices can be connected and how many devices can be granted acknowledged transmission from the gateway.

For example, the typical value for the probability of success is 0.9. We can get some values about the relation between devices and ACK percentage:

- 200 devices – 52%
- 400 devices – 11%
- 600 devices – 5%
- 800 devices – 0%

If we relax the requirement on the success probability to 80%, it is possible to increase the number of devices to 2000 for fully unacknowledged data, and to 400 for fully acknowledged data.

4.3 ALARM PLOTS

In this section we will analyze one of the possible applications of a LoRaWAN network: the implementation of alarm devices. This scenario could potentially present different problems from the one that was analyzed previously, and needs to take into account the delay metric.

4.3.1 Assumptions

Similarly to the network that was analyzed before, the network is composed by one gateway in the center of a circle with radius 7500 m and by randomly positioned, periodically transmitting end devices.

Although the alarm devices have the same features as the rest of the periodically transmitting end devices, in this simulation it is possible to see how the sending of packets from alarm devices, which will need send the packets at the same moment, SF and position, degrades the performance. The probability of collision in this special case is expected to be especially high, since multiple devices will transmit:

- Simultaneously, guaranteeing a complete overlap of the packets
- At the same SF, since they are expected to be closely positioned
- With a similar power at the gateway, since they are close to each other and thus experience the same path loss.

4.3.2 Parameters

In this case, the number of periodically transmitting end devices is fixed at 200, and an additional number of alarm devices are added and allocated in a specific circle with radius of 200 m at coordinates (1000, 1000) as shown Figure 4-5. As the previous graphic of the overview of the network distribution, the colors represent the diverse data rate where devices in the center have the highest data rate. The number of alarm devices is a parameter that varies from 1 to 20, to simulate various scenarios of simultaneous transmission. Upon a failed transmission (i.e., one that didn't get an acknowledgement), the alarm devices also try to retransmit the packet, up to 8 times.

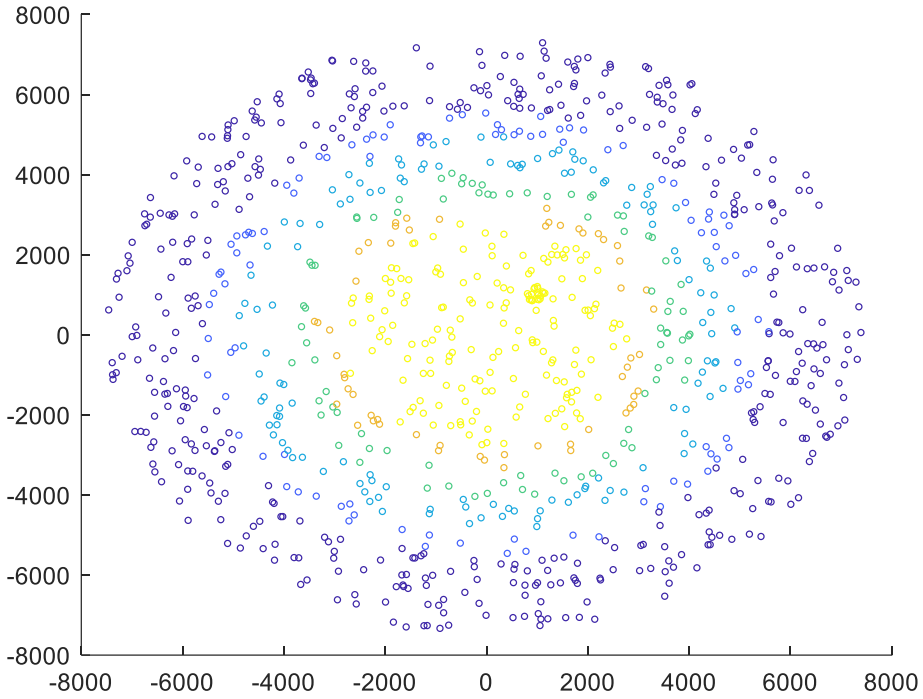

Figure 4-5 Distribution of end devices and alarms devices with their DR used

We can evaluate the performance of the alarm devices by using a variety of policies for their transmission, which are meant to mitigate the effect of interference:

- Plain: the normal behavior of alarm devices is not modified. They work as the rest of the end devices.
- Delay: before starting to send alarm packets, alarm devices wait a random time in the [0s, 10s] interval before transmitting. This policy aims at reducing the overlap between packets.
- Data rate: since all the alarm devices are positioned close to each other, the SF selected for these devices is probably always the same (as shown in Figure 4-5 for the case of SF7). This policy aims at leveraging the spreading factor orthogonality by forcing the devices to use a random spreading factor between those that will allow reception at the gateway.
- All: a combination of the Delay and Data rate policies.

To better understand the delay of alarms in the network when applying these policies, some simulations have been carried out comparing the results of different setups.

4.3.3 Delay plots

One of the most important parameters is the delay that a packet suffers during the transmission. For this reason, this characteristic is evaluated and represented in the plot in Figure 4-6.

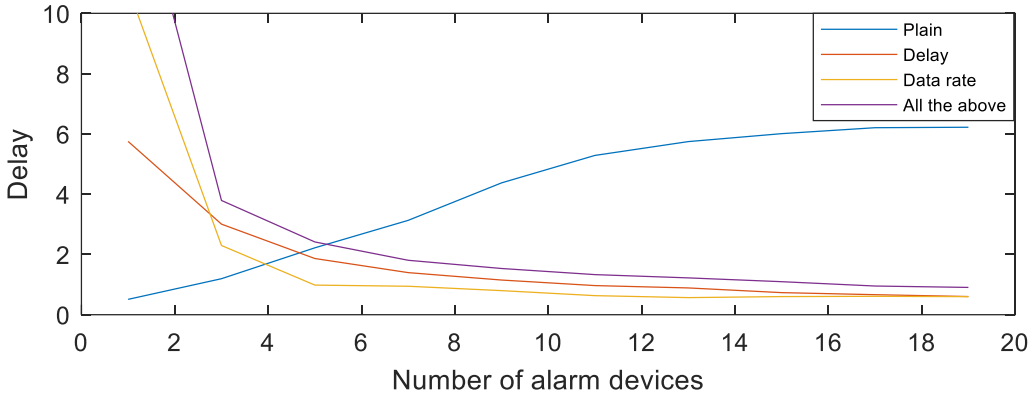

Figure 4-6 Alarm delay

For one alarm device, we can see that the best policy for delay is the plain. Instead, for the alternative policies the delay for one device is incremented of more than five seconds for the delay policy and more than ten seconds for the all and data rate policies. This last delay is so high because if a collision is produced and the SF that was randomly selected was high, this packet will have to wait a longer time to retry the transmission. In other words, the higher the SF, the longer the package will have to wait to transmit again because of the duty cycle.

As shown the Figure 4-6**¡Error! No se encuentra el origen de la referencia.**, the plain policy clearly doesn't work when multiple alarms are sent, because the delay quickly increases. This fact is caused by all packages being transmitted with the same SF, which increases the probability of collision between packets as can be seen in the Figure 4-7, representing the probability of

success at first transmission. In the case of the plain policy, collisions are guaranteed to happen if the devices pick the same channel for transmission.

On the other hand, we can see that the other policies are worse when there are less than four devices, but begin to give better results when more than five devices are involved. As the plot shows, with the increment of devices, the data rate and the delay policies are getting better, with the latter giving a better delay of around 0.5 seconds.

4.3.4 Success at first attempt plots

Other characteristic very important when a network is evaluated, is the probability of success of the packets. In this case, we are going to analyze only the success at the first attempt.

In the case of the delay, we have seen that the best policy is the data rate. On the contrary it is not the best in this performance, indicating that a good portion of the packets are lost due to interference. In the plain policy all the alarm devices are transmitting with the same SF, thus when the network has a few devices, the plain policy has a high probability of success but as shown in Figure 4-7 with multiple devices this probability is significantly reduced. Consequently, the plain policy is not a good option. Whereas the data rate policy also is affected severally by the increase of the devices, caused by the fact that higher data rates collide with higher probability, the delay and all policies maintain more or less the same probability between one device and ten devices, in other words, the probability of success at the first attempt decreases very slowly with the increase of number of alarm devices involved.

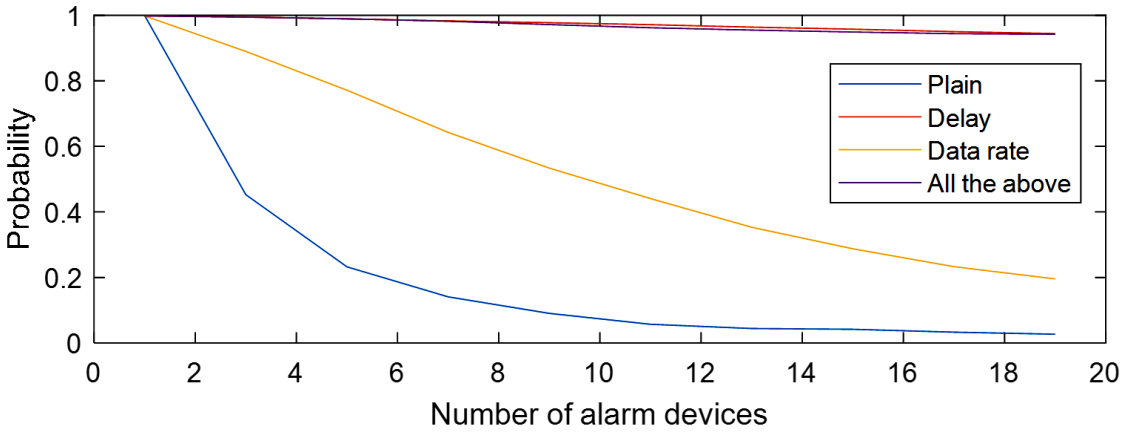

Figure 4-7 Probability of success at first attempt

4.3.5 Box plots

In this case, we use the function *boxplot* in matlab to create a box plot. On each box, the central mark indicates the median, and the bottom and top edges of the box indicate the 25th and 75th percentiles, respectively. The whiskers extend to the most extreme data points not considered outliers, and the outliers are plotted individually using the `'+'` symbol.

4.3.5.1 With one device

Firstly, we analyze the performances of the policies in the scenario where just one device is transmitting. In this case, the probability of success is not so relevant, so we focus on the delay.

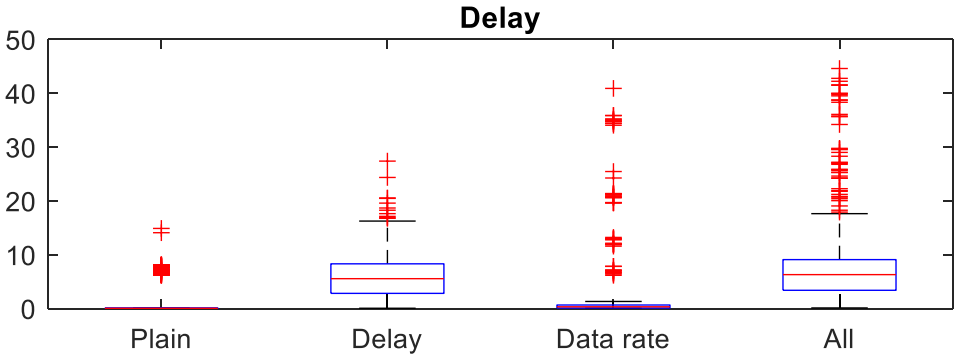

Figure 4-8 Performance with one device

For one device, the best policy is the plain because of the delay is the lowest. Although the random data rate policy's delay is also low, practically zero, it also has the potential to have big outliers, in the case of the choice of high data rates that fail the first transmission.

4.3.5.2 With multiple devices

We also analyze the performances when multiple devices are transmitting.

First, it is possible to see that the plain policy is not the best because the probability of success can be extremely low. Among the rest of the policies, the random data rate policy has more extreme outliers than the delay and all policies.

Regarding the delay, although the lowest delay is of the plain, we have seen that the probability of success is not good so analyzing the rest, we can conclude that the best policy is the delay. This fact is because in comparison with the all policy that has a similar delay and the delay is higher than the other two policies, as shown by the box plot, the delay policy has fewer outliers and the mean is lower.

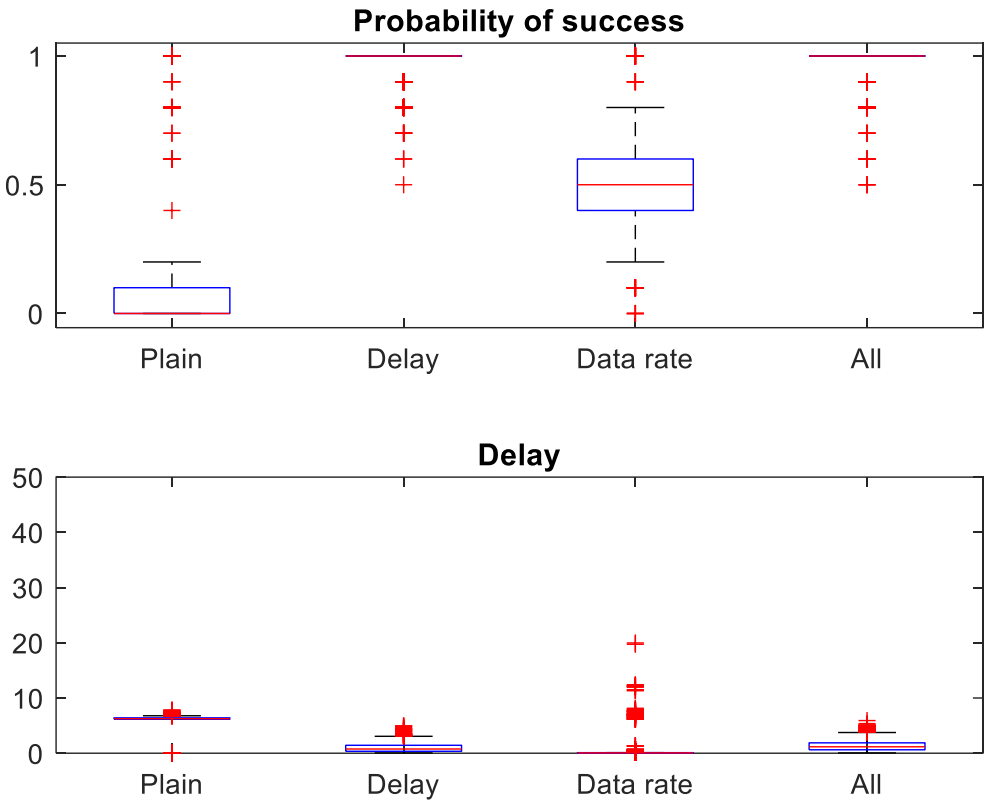

Figure 4-9 Performance with 10 devices

Finally, a scenario with 200 alarm devices simultaneously going off is shown in Figure 4-10, where we can see that the best policy in this case is the all policy, showing a consistently lower delay when compared to other policies.

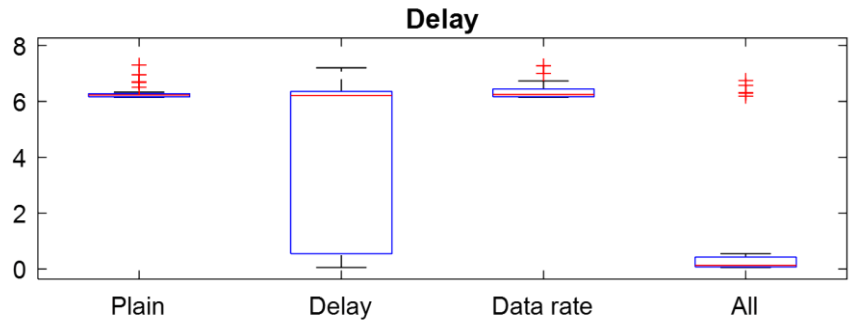

Figure 4-10 All policy performances

5 CONCLUSIONS

In this work, we worked with one of the most prominent LPWAN technologies used for IoT applications. After an introduction where we show the requirements needed nowadays for massive IoT connectivity, were analysed the most relevant alternatives of LPWAN technologies, as SigFox, Weightless and LoRa.

In this thesis we have chosen the LoRa and LoRaWAN technologies, in order to analyze the performance of a network for confirmed and synchronous traffic. For this reason, a review of the main characteristics, ranges and capabilities was provided.

Following the specification proposed by the LoRa Alliance and focusing on the basic requirements of IoT, LoRa network architecture and components of a LoRaWAN network employing the LoRa modulation have been described.

Then, making a brief instruction about ns-3 where its organization is described, together with the main elements and the steps of a simulation, we modified an ns–3 module to study the performance of a LoRa network.

The first simulation results show how the probability of success is decreased with the increase of devices in the network, and with the presence of acknowledged frames. More specifically, performance degrades much more quickly when traffic is heavy, especially when a high percentage of devices require an acknowledgement. Next, a new scenario with synchronous alarm devices was simulated, and three proposed policies were implemented to mitigate the problem of collision of simultaneously transmitted packets. The policies were shown to achieve a better performance based heavily on the number of devices which take part in the simultaneous transmission.

5.1 FUTURE DEVELOPMENTS

In the future, the analyzed policies could be developed to get further improvements in performance and simulate even more different strategies.

Various choices of distributions for the spreading factor in the Data Rate policy, giving a priority to low SFs, could be simulated to gain insight on the optimum distribution. Another possible area of improvement is the distribution of delay and the maximum value of delay, since in this thesis only the performance using a delay of 10 seconds was analysed.

REFERENCES

[1] A. Boulogeorgos, Panagiotis D. Diamantoulakis, George K. Karagiannidis, “Low Power Wide Area Networks (LPWANs) for Internet of Things (IoT) Applications: Research Challenges and Future Trends.”

[2] Ericsson, “Cellular networks for Massive IoT,” 2016.

[3] M. Centerano, L. Vangelista, A. Zanella, M. Zorzi, “Long-Range Communications in Unlicensed Bands: the Rising Stars in the IoT and Smart City Scenarios,” IEEE Wireless Comunications.

[4] “About 3GPP.” [Online]. Available: http://www.3gpp.org/about-3gpp.

[5] Keith E. Nolan, Wael Guibene, Mark Y. Kelly, “An evaluation of low power wide area network technologies for the Internet of Things.” IEEE.

[6] P. Pardal, Wide Area Networks for IoT applications,” 2017.

[7] D. Magrin, “Network level performances of a LoRa system,” Master Thesis. Università degli Studi di Padova, 2017.

[8] U. R. MartinBor, Jonh Vidle, “LoRa for the internet of Things.” EWSN ’16 Proceedings of the 2016 International Conference on Embedded Wireless Systems and Networks, pp. 361–366, 2016.

[9] C. Goursaud, J.M. Gorce, “Dedicated networks for IoT: PHY/MAC state of the art and challenges.” EAI endorsed transactions on Internet of Things, 2015.

[10] N. Sornin, M. Luis, T. Eirich, T. Kramp, O.Hersent, “LoRa Specification.” LoRa Alliance, 2015.

[11] K. Mikhaylov, J. Petäjäjärvi, T. Hänninen, “Analysis of the Capacity and Scalability of the LoRa Wide Area Network Technology,” University of Oulu, Finland.

[12] “NS-3 Manual.” [Online]. Available: https://www.nsnam.org/docs/manual/html/index.html.

[13] “NS-3 Documentation.” [Online]. Available: https://www.nsnam.org/docs/release/3.10/tutorial/html/conceptual-overview.html#key-abstractions.

[14] K. Wehrle, M.Günes, J. Gross, "*Modeling and Tools for network simulations"*. 2010.

[15] P. L’Ecuyer, R. Simard, E. J. Chen, W. D. Kelton, “An object-oriented random-number package with many long streams and substreams.” Oper. Res., vol. 50, no. 6, pp. 1073–1075.

[16] M. Centenaro, D. Magrin, L. Vangelista, “Performance Evaluation of LoRa Networks in a Smart City Scenario.”