Università degli Studi di Padova

Department of Information Engineering

Master Thesis in Telecommunication Engineering

Network level performances of a LoRa system

Supervisor
Lorenzo Vangelista
Università di Padova

Co-supervisor
Marco Centenaro
Università di Padova

Master Candidate
Davide Magrin

Abstract

The demand for connected devices, according to the Internet of Things (IoT) paradigm, is expected to grow considerably in the immediate future. Various standards are currently contending to gain an edge over the competition and provide the massive connectivity that will be required by a world in which everyday objects are expected to communicate with each other. Among these standards, Low-Power Wide Area Networks (LPWANs) are continuously gaining momentum, mainly thanks to their ability to provide long-range coverage to devices, exploiting license-free frequency bands. The focus of this thesis is on one of the most prominent LPWAN technologies: LoRa™.

First, this thesis establishes a series of models that cover various aspects of a LoRa network. Then, a new Network Simulator 3 (NS3) module is introduced to simulate a LoRa-based IoT network in a typical urban scenario. Finally, the performance of the LoRa system is evaluated and analyzed.

Sommario

Si prevede che la presenza di dispositivi connessi, secondo il paradigma dell'Internet delle Cose (IoT), aumenterà sostanzialmente nell'immediato futuro. Diversi standard sono attualmente in competizione per aggiudicarsi la maggioranza del mercato e fornire la connettività su larga scala che è richiesta in un mondo dove molti oggetti della vita quotidiana saranno in grado di comunicare tra loro. Tra questi standard, le Low Power Wide Area Networks (LPWAN) sono in forte crescita, soprattutto grazie alla loro connettività a lungo raggio sfruttando bande di frequenza libere. Questa Tesi si focalizzerà su una delle tecnologie LPWAN predominanti: LoRa™.

Prima di tutto verranno introdotti dei modelli utili a rappresentare le caratteristiche di una rete LoRa. Successivamente, verrà presentato un nuovo modulo per includere la tecnologia LoRa nel simulatore di rete Network Simulator 3 (NS3). Infine, tramite tale modulo, si studieranno le prestazioni di una rete LoRa in ambito urbano.

Contents

Listing of figures

Listing of tables

Listing of acronyms

3GPP 3rd Generation Partnership Project

6LoWPAN IPv6 over Low power Wireless Personal Area Networks

ABP Activation By Personalization

ACK Acknowledgement

ADR Adaptive Data Rate

BLE Bluetooth Low Energy

BLER Block Error Ratio

CAD Carrier Activity Detection

CIoT Cellular IoT

CS Continuous Simulation

CSS Chirp Spread Spectrum

ECC CEPT's Electronic Communications Committee

DDR Dynamic Data Rate

DER Data Extraction Rate

DES Discrete Event Simulation

DL Downlink

DSSS Direct Sequence Spread Spectrum

ED End Device

EC-GSM Extended Coverage GSM

ERP Effective Radiated Power

EWL External Wall Loss

ETSI European Telecommunications Standard Institute

EIRP Effective Isotropic Radiated Power

ERP Effective Radiated Power

FEC Forward Error Correction

FFT Fast Fourier Transform

GW Gateway

GPL General Public License

IoT Internet of Things

ISI Inter-Symbol Interference

ISM Industrial, Scientific, and Medical

ITU International Telecommunication Union

LBT Listen-Before-Talk

LPWAN Low-Power Wide Area Network

LR-WPAN Low-Rate Wireless Personal Area Network

LTE Long Term Evolution

MIC Message Integrity Code

MAC Medium Access Control

MAR Mobile Autonomous Reporting

MFSK Multiple Frequency Shift Keying

MCS Modulation and Coding Scheme

NACK Not-Acknowledgement

NS3 Network Simulator 3

NS Network Server

OTAA Over-The-Air Activation

PHY Physical Layer

PRNG Pseudo Random Number Generator

ROI Region of Interest

SDR Software Defined Radio

SINR Signal to Interference plus Noise Ratio

SISO Single Input Single Output

SF Spreading Factor

ToA Time on Air

TTI Transmission Time Interval

UL Uplink

UNB Ultra Narrow Band

1 Introduction

More and more everyday objects are being connected to the Internet, gradually building the future described by the Internet of Things (IoT) paradigm. The smart device market is expected to grow considerably, with estimates stating that 20 billion IoT devices or more will be active by 2020 [1], and consequently bring a projected annual economic impact in the range of $2.7 trillion to $6.2 trillion by 2025 [2]. Areas of application for this new connectivity and data gathering paradigm include the fields of health care, transportation, smart homes, agriculture, manufacturing and urban infrastructure management among many others, with health care and manufacturing expected to be the dominant markets [2].

There are various architectures attempting to represent how IoT devices will operate. However, the most basic model can be identified in a stack consisting of three components, each one corresponding to a different high-level task, as depicted in Figure 1.1:

1. a Perception layer collects data from sensors and controls actuators;

2. a Network layer interconnects devices so that they can share information between themselves or with a centralized data sink, and

3. an Application layer stores, interprets and makes use of the collected data.

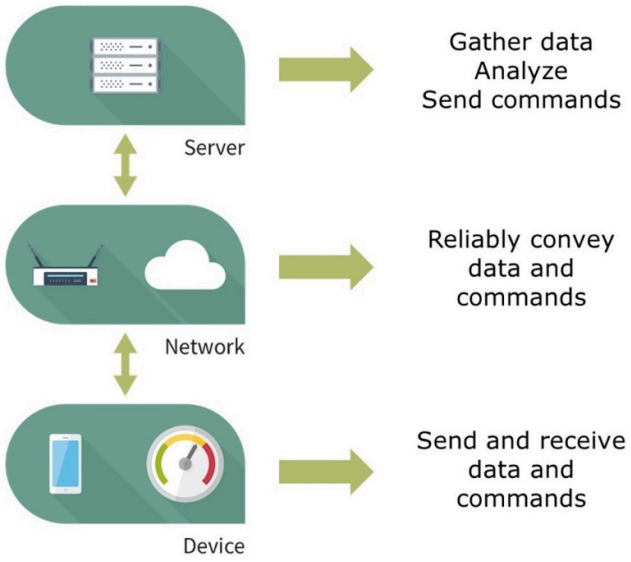


Figure 1.1: The IoT three-layer structure.

The IoT scenario poses some key challenges that need to be tackled and solved by any technology operating at the Network layer:

- **Scalability**: it is forecast that device density will be in the order of 60000 devices per km^2 [3]. IoT networks will need to support modulations and medium access schemes that allow for such a densely populated wireless environment, and dynamically adapt network parameters to achieve the best possible efficiency in the face of frequent network congestion and packet collision.

- **Device cost**: radio chips will need to be as cheap as possible, to help vendors gain an edge in the market by reducing the price for a device. Subscription costs to access the network are also required to be as low as possible.

- **Device battery life**: the majority of IoT devices will run on batteries, and a long autonomy is necessary to shrink network maintenance costs. A commonly accepted target figure for the battery life of a sensing and reporting device is in the order of 5 to 10 years on 2 AA batteries, depending on the frequency of transmission.

- **Device computing power**: IoT devices are expected to have very basic CPUs. This restriction limits the complexity of network protocols and modulation that needs to be used by these devices.
- **Deep indoor coverage**: devices are expected to be able to communicate even when heavy shadowing is involved. This holds true especially for critical applications, where requirements on the rate of successful message delivery will be very strict.

Even though the above requirements must be met, the following two aspects are not considered to be crucial for the performance of IoT devices:

- **Throughput**: IoT devices are not expected to support high throughput. Instead, data is expected to be shared infrequently and in small amounts.
- **Persistent connection**: the vast majority of IoT devices will not be required to be constantly active. This, for example, leaves room for a network protocol to leverage the device's sleep mode to decrease battery consumption at the cost of a delay in transmission.

The capability of an IoT network protocol to fulfill the aforementioned requirements needs to be carefully investigated before a massive deployment can be implemented. Currently, a debate is going on about the effective performance of many different network standards. One particular architecture for IoT networks, the Low-Power Wide Area Network (LPWAN) paradigm, is still in the evaluation phase in the research community: the objective is to understand whether these networks are a viable solution for the deployment of massive IoT, and whether they will be able to compete with other standards. In particular, this thesis aims at evaluating the performance of one of the most prominent LPWAN technology, LoRaWAN™, in a typical urban scenario. Comprehensive and accurate system-level simulations of LoRa networks that consider a number of end-nodes which are deployed in a realistic propagation scenario, with streets and buildings, are still missing. It is in this context that this thesis is presenting novel results, demonstrating that a LoRaWAN provides a higher throughput than a typical ALOHA-based protocol, without increasing the Medium Access Control (MAC) complexity. In order to properly simulate a LoRaWAN network, a model for a

LoRa network is first proposed and then implemented to develop a new module in one of the most accurate system-level network simulators that are currently available: ns–3 [4]. Different simulations are then performed with this new tool in order to evaluate throughput, coverage and many other important metrics that can be used to design an efficient network employing LoRa technology.

The rest of this thesis is organized as follows:

- Chapter 2 provides an overview of different solutions for IoT connectivity that are currently available or in the development phase, before focusing on the PHY and MAC layers of the LoRa technology.
- Chapter 3 explains how the previously described aspects of a LoRa link and network are modeled and simplified to perform a system level simulation of this new technology.
- Chapter 4 introduces Network Simulator 3, and then focuses on the structure of the new `lora` module that has been developed.
- Chapter 5 shows and discusses the simulation results.
- Finally, Chapter 6 draws the conclusions of this work and identifies possible future directions on the same topic.

2

IoT Technologies

As explained in Chapter 1, the IoT paradigm actually comprises three layers that deal with sensing, network and data management. Since this work will analyze one of the proposed solutions for the Network portion of the stack, the following sections provide an overview of the different technologies that have been designed so far to connect IoT devices. After this introduction and comparison of rival technologies, the LoRa modulation and the LoRaWAN standard will be covered, in order to describe the mechanisms that will be modeled and simulated in the following chapters.

2.1 Solutions for IoT Connectivity

One can identify three main categories of competitors, which vary in range, throughput and cost. Not necessarily only one of these approaches will survive, since each one has some strengths and weaknesses when compared to the others. Nevertheless, it is clear that there currently is a competition between multiple architectures, and that one will prevail and end up providing the bulk of the connectivity to IoT devices.

2.1.1 Low-Rate Wireless Personal Area Networks

Low-Rate Wireless Personal Area Network (LR-WPAN) technologies create small networks, typically covering and interconnecting the devices owned by an individual or operating in a house. These standards provide low data rates and short range communication, in order to focus on efficient battery use.

The IEEE 802.15.4 specification for the PHY and MAC layers provides a starting point for many different solutions that aim at completing the standard with the upper layers, like IPv6 over Low power Wireless Personal Area Networks (6LoWPAN), Z-Wave and Thread. At the network level, these solutions assume a mesh topology, thus featuring multi-hop connections: while increasing the robustness of the connection is an advantage, using nodes as relays limits their ability to go into sleep mode to save battery power, and routing requires additional computational work. The range of a single hop for these devices is around 10 m, with a raw data rate between 20 and 250 kbit/s depending on the band that is used. Another notable LR-WPAN standard is Bluetooth: with the recent introduction of Bluetooth Low Energy (BLE), this technology is now able to provide range of 100 m, and application-level data rates of 270 kbit/s, and is thus suitable for applications similar to those of standards based on IEEE 802.15.4.

2.1.2 Cellular IoT

Cellular IoT (CIoT) standards will operate in licensed bands and leverage the already existing cellular network coverage to provide internet access to IoT devices: the fact that the infrastructures for the network are already installed is a great benefit and will make deployment time very short.

Currently, three different standards have been proposed [5]: EC-GSM, LTE-M and NB-IoT. EC-GSM is designed to leverage and improve on legacy EDGE and GPRS systems to provide better coverage and range, with limited power requirements. LTE-M will integrate with LTE to make use of its capacity and performance and bring new power saving options to increase device battery life. Finally, the new NB-IoT standard will focus on ultra-low-end IoT applications, once again leveraging the existing LTE infrastructures [5]. Future 5G networks are also expected to provide connectivity to IoT devices by design.

2.1.3 Low Power Wide Area Networks

LPWANs have recently been emerging as an alternative to LR-WPANs and CIoT, mainly thanks to the range limitations of LR-WPANs and to the fact that CIoT is still in a very early stage of deployment. These networks provide wireless connectivity using a star topology and long-range transmission in the unlicensed sub-GHz frequency bands [6]. The other great benefit brought by LPWANs is an increased power efficiency: many of these technologies so far have made the claim of being able to sustain a device for 10 years on a couple AA batteries.

One of the main competitors among standards for this architecture is Sigfox, which employs an Ultra Narrow Band (UNB) modulation, capable of sending a 12 bytes payload in 6 seconds using a 100 Hz band. Due to band regulations, Sigfox currently allows users to send up to 140 messages per day. Messages sent by devices are then picked up by a gateway, which is connected to a centralized Sigfox server. From a device's perspective, access to the wireless medium is not regimented, making Sigfox an ALOHA-like random access scheme. Sigfox networks are expected to provide massive access to devices based on the fact that signals are very narrowband and that transmission frequencies are randomly chosen. Finally, three copies of every packet are sent, at different random frequencies, in order to reduce the probability of losing a packet because of collisions or frequency selective fading. Sigfox also limits the number of packets that can be sent downlink from the network to a device to 4 messages per day.

Contrary to Sigfox, LoRa is a technology that exploits a new spread spectrum Physical Layer (PHY) design that enables a higher receiver sensitivity in order to trade data rate for coverage, decreasing the former to increase the latter. LoRa and LoRaWAN are, respectively, a proprietary modulation developed and owned by Semtech Corporation [7] and a network standard, focused on leveraging useful properties of the LoRa modulation, proposed by the LoRa Alliance [8]. The LoRa modulation allows for very good receiver sensitivities at a contained chip cost, thus achieving long range transmissions (up to 13 miles in a rural environment) at the price of a reduced data rate, in the $0.3 - 50$ kbps range. At the same time, the LoRaWAN standard that allows multiple LoRa devices to communicate together aims at shifting the burden of administering the network towards a central control point. This allows devices to be as simple as possible, and gives a

central coordinator the power of easily tuning each device's parameters in order to accommodate new nodes in the network. Since the work of this thesis is focused on this technology, the following sections will explore how the LoRa modulation and LoRaWAN were designed to meet the IoT requirements listed in Chapter 1.

2.2 The LoRa Modulation

LoRa is a proprietary PHY layer technology, based on the Chirp Spread Spectrum (CSS) modulation technique. Because of the fact that this technology is patented, and no clear description of the modulation is available. Some pieces of information can be found in semi-official documents from Semtech and the LoRa Alliance, such as [9, 10, 11]. This void in the documentation, though, has been filled by a few researchers and hobbyists that analyzed and successfully reverse engineered the modulation. The most comprehensive example is Matt Knight's work [12].

2.2.1 LoRa's Chirp Spread Spectrum Implementation

The idea behind CSS is that a sinusoidal signal of linearly varying frequency and fixed duration, called *chirp*, can be employed to "spread" information over a wider spectrum than it would normally need to occupy. This uniform distribution of a symbol over a larger bandwidth provides resistance to frequency-selective noise and interferers, at the price of a lower spectral efficiency. Using some additional precautions, CSS can also be more resilient to multi-path interference and the Doppler effect than other more conventional modulations. Let's assume that the available frequency band for transmission is $\mathcal{B} = [f_0, f_1]$. A chirp can be constructed so that it increases linearly in frequency from a starting frequency $f_s \in \mathcal{B}$ to that same frequency, wrapping around from f_1 to f_0 when hitting the end of the available band. In LoRa, a chirp's starting frequency inside the available bandwidth seems to be the used to represent a symbol [13]. The number of bits that LoRa encodes in a symbol is a tunable parameter, called SF. This means that a chirp using spreading factor SF represents 2^{SF} bits using a symbol, and that there are $M = 2^{SF}$ possible starting frequencies for a chirp. A transmission's spreading factor is also used to determine the duration of a symbol, according to the following expression:

Table 2.1: SNR values for different spreading factors.

SF	SNR
7	-7.5 dB
8	-10 dB
9	-12.5 dB
10	-15 dB
11	-17.5 dB
12	-20 dB

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

This means that, assuming the modulation is using a fixed bandwidth, an increase of spreading factor of 1 will yield symbols that last twice the duration. Analogously, a bigger bandwidth increases the rate at which chirps are transmitted, and consequently the bitrate of the modulation. An increase in the transmit time for a chirp (i.e., a symbol) gives the message an higher robustness to interference or noise. On the other hand, this effect may be partially balanced by the fact that for higher spreading factors the number of possible symbols increases, making the occurrence of symbol errors more likely: the reason for this is that achieving synchronicity between the receiver and the signal especially critical when low data rates are employed. Another disadvantage of transmitting longer messages is the increased probability of collisions.

Because of the reasons above, the choice of SF affects receiver sensitivity, which is defined as:

$$S = -174 + 10\log_{10}(B) + \text{NF} + \text{SNR} \quad \text{dB}, \tag{2.2}$$

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. SNR values for different spreading factors are represented in Table 2.1, where it can be seen that increasing the spreading factor allows for a better sensitivity.

Given Eq. (2.1), we can now get the bitrate for a certain pair of SF and B

Table 2.2: Bitrate [bits/s] for a range of spreading factors and bandwidths.

SF	125 kHz	250 kHz	500 kHz
7	6835	13671	27343
8	3906	7812	15625
9	2197	4396	8793
10	1220	2441	4882
11	671	1342	2685
12	366	732	1464

using a simple computation:

$$R_b = \frac{SF}{T_s}. \tag{2.3}$$

The bitrates for a range of spreading factors and bandwidths can be found in Table 2.2.

2.2.2 LoRa Physical Layer Packets

An example of a LoRa packet can be seen in Figure 2.1, which shows a spectrogram representation with time on the horizontal axis and frequency in the vertical axis. It can be seen that a PHY layer LoRa message consists in the chirp signal sweeping the frequency band. After some repetitions of this frequency sweep that constitute a preamble (whose minimum length is of 4.25 chirps), data is encoded in the signal as instantaneous changes in the frequency of the chirp, or lack thereof. A decoding process is proposed in [13], and it consists in first "de-chirping" the signal, and then taking a Fast Fourier Transform (FFT) of the signal, with a number of bins equal to the number of symbols M that corresponds to the used spreading factor. Figure 2.2 shows the de-chirped version of a LoRa transmission, again with time on the horizontal axis and frequency on the vertical one: the signal can now be interpreted as if it were modulated using Multiple Frequency Shift Keying (MFSK). By taking multiple overlapping FFTs and looking at the bin with the highest power content we can now detect the symbol in each time frame.

Figure 2.1: Spectrogram representation of a LoRa signal [13].

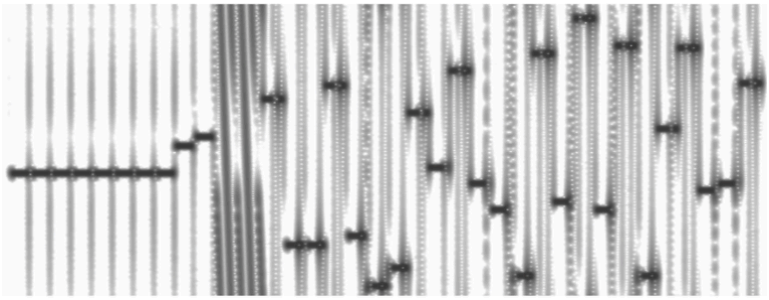


Figure 2.2: De-chirped version of a LoRa signal.

Aside from the modulation itself, LoRa also specifies a set of encoding operations that are applied before modulation and transmission:

- **Data whitening** is used in order to shrink the probability of long equal bit runs occurring in the data. Whitening also helps distributing information across the whole bandwidth of the radio channel. It should be noted that [13] found the whitening sequences specified in [14] to be different than the version that is actually implemented in the chips, but was still able to find out the right whitening sequence.

- **Forward Error Correction (FEC)** is implemented as an Hamming Code. The code's information word length is fixed at 4 bits, and the length of the codeword is a tunable parameter in the $[5, 8]$ bits range. The code rate for a LoRa packet, then, is $\mathcal{C} \in \{4/5,\ 4/6,\ 4/7,\ 4/8\}$.

- **Interleaving** scrambles the output of FEC to make the code more resistant against bursts of error. Once again, [13] found that the documents that are available online differ from the chip implementation. Through reverse

engineering, it was found that the modulation uses a diagonal interleaver, with the two most significant bits reversed.

- **Grey mapping** is finally used to map a block of SF bits into one of the M symbols in the constellation, while making sure that two adjacent symbols differ by at most 1 bit, in order to increase the channel code's chances to correct possible errors.

The on-air time of a packet can be computed with the following formula [9]:

$$t_{\text{packet}} = t_{\text{preamble}} + t_{\text{payload}}, \tag{2.4}$$

where t_{preamble} is the time it takes to transmit the preamble, and t_{payload} is the time to transmit the actual data. These two entities have the following expressions:

$$t_{\text{preamble}} = (n_{\text{preamble}} + 4.25) \cdot t_s, \tag{2.5}$$

$$t_{\text{payload}} = n_{payload} \cdot t_s, \tag{2.6}$$

where n_{preamble} is a configurable parameter that affects the number of symbols in the preamble (and thus the probability that a receiver will detect an incoming packet, at the cost of a longer time on air).

The computation of n_{payload} is instead more complicated, since it depends on many different parameters:

- PL is the number of payload bytes;
- H can be either 0 when the PHY header is disabled, or 1 when it is set to enabled. The PHY header is used to carry information about the packet length, enabling variable payload sizes. This information can be omitted to save on-air time if the transmitter and receiver both know the duration of a packet.
- DE can be either 0 when the low data rate optimization is disabled, or 1 when it is enabled. Low data rate optimization is a measure used to counter clock drift when sending very long symbols, and to achieve correct time synchronization between transmitter and receiver.

- CR is the number of added parity bits.

Given the above parameters, the number of payload symbols becomes:

$$n_{\text{payload}} = 8 + \max\left(\left\lceil \frac{8\text{PL} - 4\text{SF} + 44 - 20\text{H}}{4\left(\text{SF} - 2\text{DE}\right)} \right\rceil \left(\text{CR} + 4\right), 0\right) \tag{2.7}$$

2.2.3 Spreading Factor Orthogonality

One very powerful feature of the LoRa modulation is that different spreading factors are pseudo-orthogonal, even when the same center frequency and bandwidth settings are used. This allows a receiver to correctly detect a packet using spreading factor i even if it is overlapping in time with another transmission employing spreading factor j, as long as $i \neq j$ 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. This pseudo-orthogonality between different packets allows a network employing LoRa devices to exploit different spreading factors to achieve an higher throughput with respect to more traditional modulation schemes, in which a collision can cause the incorrect reception of both the intended packet and the interferer. While the exact figure for the isolation margin is never explicitly stated in Semtech documents, in [15] this was investigated and some estimates were made based on a model of the LoRa simulation.

2.2.4 Main Semtech Chips and Independent Implementations

Since the technology is proprietary, commercially available chips implementing the LoRa modulation chain are currently only available from Semtech Corporation. There are two main kinds of LoRa radio chips: the SX1272 and SX1273 are the most basic chips, intended for simple LoRa devices, while the SX1301 is able to decode multiple packets on different frequencies simultaneously. This chip is intended to be used in aggregators that can pick up transmissions from a whole network made of simpler LoRa devices. The scheme for a SX1301 receiver can be seen in Figure 2.3: the chip has 8 built-in configurable Dynamic Data Rate (DDR) *receive paths*. These blocks are able to work concurrently in order to decode different overlapping transmissions using any spreading factor and center

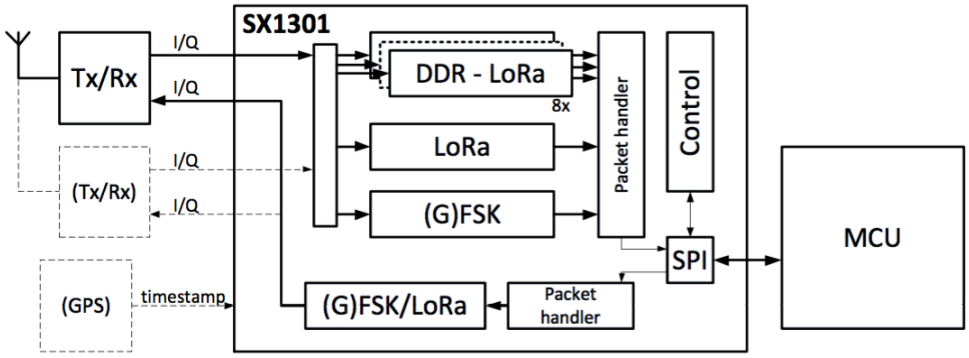


Figure 2.3: Logic scheme for the SX1301 chip.

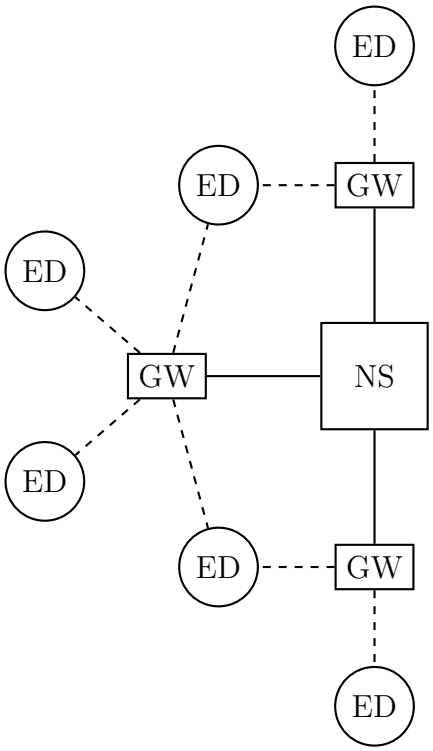
frequency [10], leveraging the pseudo-orthogonality between different spreading factors to distinguish between packets arriving at the antenna simultaneously and decode each one correctly.

Aside from the official vendor's chips, some Software Defined Radio (SDR) projects devoted to implementing an open LoRa PHY layer recently came to life: the most notable examples are [16] by Matt Knight and [17] by Josh Blum. Both implementations leverage the software development toolkit GNU Radio [18]. These open implementations are often created by researchers and hobbyists in order to explore the modulation: these tools can be used to identify ways to improve the modulation and to expose some of its defects or security weaknesses.

2.3 The LoRaWAN Standard

While the LoRa PHY layer is proprietary, the rest of the protocol, known as LoRaWAN, is open and described in [11] by the LoRa Alliance, a group of vendors and research institutions that are interested in spreading and leveraging LoRa technology. What follows is a brief overview of the main components of a LoRaWAN, and of the frequencies it was decided the networks will operate in different parts of the world.

Figure 2.4: Sample topology of a LoRa network.



2.3.1 Topology and Device Classes

LoRaWAN networks are laid out in a star-of-stars topology, in which *End Devices (EDs)* send/receive messages wirelessly to/from one or more *Gateways (GWs)*, which in turn relay them to a centralized *Network Server (NS)* via an high-throughput and reliable link. This topology, as represented in Figure 2.4, actually allows one ED to deliver messages to more than one gateway. In fact, EDs are not explicitly paired with a single gateway: messages are simply sent by the devices on the wireless channel, assuming that at least one gateway will receive them and forward them to the NS. It is then the centralized system that has the responsibility of filtering duplicates and picking the most appropriate gateway through which to send downlink messages to that device. In order to increase the network more robust to interference, multiple logical channels are defined for the whole network, and devices needing to transmit a packet are required to pick a channel in a pseudo-random fashion.

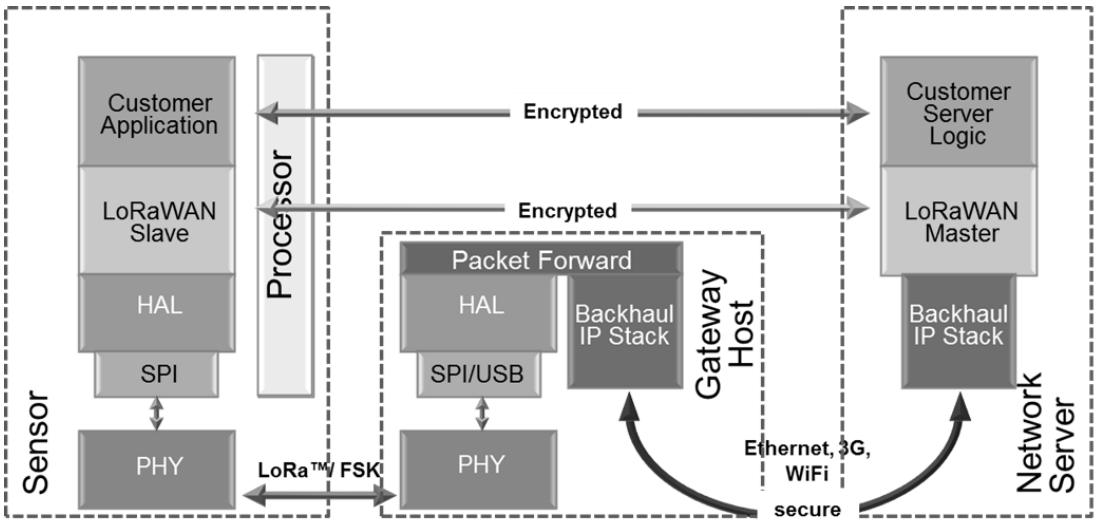


Figure 2.5: Protocol stacks of the various devices in a LoRaWAN.

The protocol stack of EDs, GWs and of the NS is represented in Figure 2.5. While the ED and NS stacks have an application layer, gateways are only tasked with forwarding messages between the sensor (i.e., the EDs) and the NS, and are thus totally transparent to the end devices' application, which is logically connected directly to the one on the NS.

According to [11], a LoRa ED can behave according to one of the following models:

1. *Class A* is the default operation mode of LoRa end devices. Each end device transmits packets coming from upper layers on the wireless channel in a totally asynchronous fashion, thus implementing an Aloha Medium Access Control (MAC). After each uplink transmission, at most two reception windows are opened by the node, waiting for any command or data packet returned by the NS. The first window is opened at the same channel as the node's uplink communication, while the second window is opened on a different sub-band (previously agreed upon with the NS, and modifiable through MAC commands) in order to increase the resilience against channel fluctuations. This class is expected to be implemented by devices whose energy budget is limited and therefore should keep the radio transceiver off as long as possible.

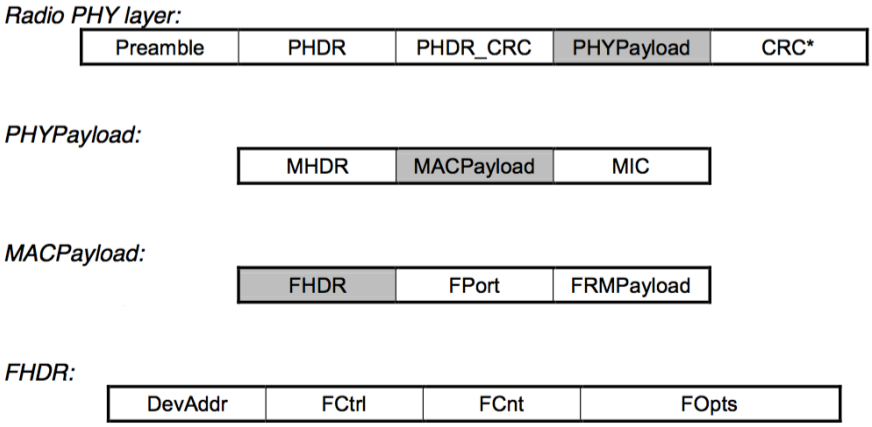


Figure 2.6: Packet structure of a LoRaWAN message [11].

2. *Class B* end devices are synchronized with the NS by means of beacon packets which are broadcast by Class B gateways. Thanks to the beacon mechanism, they can receive downlink data or command packets in specific time windows, irrespective of the uplink traffic. Class B is intended for end devices that need to receive commands from a remote controller, e.g., switches or actuators.

3. Finally, *Class C* is defined for end devices without strict energy constraints (e.g., devices that are connected to the power grid), which can hence keep the transceiver constantly on, waiting for downlink messages. This class is expected to be useful to devices operating an actuator, and having stricter delay requirements.

2.3.2 Packet Structure and MAC Commands

Additionally to the topology of the network, [11] also describes the communication protocol. This includes the format of PHY and MAC layer packets, a set of network parameters, like the SF and channel frequencies used by an end device, and the MAC commands which must be used to tune the aforementioned parameters. The packet structure of a LoRaWAN message can be seen in Figure 2.6: at the PHY layer, the packet is made of a preamble as seen in Figure 2.1, an header, a payload and two CRC codes to protect header and payload. Inside the PHY

payload, the MAC header contains information about the version of the standard the device is using and about the type of the message:

- Join packets are the first packet that is sent by a device attempting to enter a network.
- Data messages can be either uplink or downlink. Additionally, it is stated whether the message requires an acknowledgement or not.
- Proprietary messages can be used to implement non-standard message formats that can be used between devices having a common understanding of some proprietary extensions.

The MAC payload, instead, contains itself a Frame payload, a Frame port and a Frame header. The Frame payload typically contains data coming from the Application Layer, and the Frame port field is used to identify which application the message is intended for. Some port numbers are reserved for standardized application extensions that will be created in the future. The Frame header, instead, contains various fields that are tied to different aspects of a LoRa network. First of all, a 4-byte short device address is used to identify a device in a network. One byte, called Frame Control field, is intended to house the Acknowledgement (ACK) and Frame pending bits. Additionally to these, 2 more bits are used to implement the Adaptive Data Rate (ADR) feature: the NS has the ability to ask an ED to modify the spreading factor it employs for transmission. The ADR algorithm may choose to increase the spreading factor of a device whose SNR margin is too low, or decrease it in case the device's transmissions arrive consistently well above sensitivity. In other words, the mechanism needs to ensure the reliability of a device's link (typically by using high SF values) while minimizing the on-air time of the device's packets (by using lower SFs) to avoid collisions. It should be noted that the ADR algorithm is not standardized: this means that implementations that weight efficiency and complexity in different ways are possible, and open to be compared. If the ADR bit is set, the device knows that its data rate is being controlled by the NS, and that it will find further instructions among the MAC commands contained in the FOpts field.

Various MAC commands that can be exchanged between the EDs and the NS enable the following features:

- Link checking: a device can ask its link margin to the NS, which replies with the power that was received at the gateway.
- ADR requests and replies.
- Duty Cycle: the NS can set limitations to a device's aggregated transmit time.
- Setting of parameters regarding the receive windows.
- Device status: the NS can ask a device for its battery level and demodulation margin.
- Creation of new radio channels.

Additionally to the features above, 128 more command identifiers are reserved for proprietary network command extensions.

2.3.3 Encryption and Device Activation

According to [11], if a data frame carries a payload, the Frame payload must be encrypted. The encryption scheme employed by LoRaWAN is based on the algorithm that is also used by the IEEE 802.15.4 standard [19], using an AES cipher and a Message Integrity Code (MIC) to prevent attackers from tampering with the encrypted message without the receiver noticing. The key used in the AES procedure can be either a network-wide or an application-specific key, depending on the Frame port the message refers to. Both these keys are first obtained by an ED during its *activation*, a procedure thanks to which a device first accesses the network, also obtaining a 32 bit address that uniquely identifies it inside the network and an application identifier. The LoRaWAN standard provides two ways in which a device may perform activation in a LoRaWAN: Over-The-Air Activation (OTAA) allows a device to be activated by exchanging a series of messages with the NS in order to obtain a Network key over the air in a secure way, while Activation By Personalization (ABP) implies that the device comes preconfigured with an address and both the Network and the Application keys, thus being able to directly accessing a predefined network without exchanging data with the NS in join procedures.

2.3.4 Frequency Bands

In [11] three regions are defined in which LoRaWANs are expected to operate at fixed frequencies, based on local regulations in Europe, China and the United States as shown in Table 2.3. For each one of these regions, the standard mandates customized parameters that define the preamble, channel frequencies, allowed spreading factors, maximum payload size, receive windows and Join procedures to make sure that LoRaWAN always complies with the local law.

Table 2.3: Frequency bands for various regions [11].

Region	Frequency band [MHz]
Europe	868–870
US	902–928
China	779–787

It can be noted that all chosen frequency bands are in the 780–930 MHz range: in fact, for a LPWAN needing to focus on achieving the widest range possible this set of frequencies is preferable to the 2.4 GHz and 5 GHz ISM bands that are leveraged by IEEE 802.11 standards, since attenuation is lower.

2.3.5 Notable LoRaWAN Implementations

Some fully working LoRa networks have already been successfully deployed. Senet is an American company that offers a subscription based service, allowing clients to connect to a wide LoRa network covering more than 225 cities in the United States, including Los Angeles, New York City, Washington D.C., Chicago, Philadelphia, Dallas, Seattle, San Diego, Atlanta, Denver, San Francisco, Boston and San Jose, for a total covered population of 50 million people. Typical applications include irrigation, parking and water monitoring. Another notable implementation of a LoRaWAN is The Things Network, a crowdsourced solution for the IoT leveraging LoRa technology, thanks to which anyone can install their own gateway an connect it to the organization's central network server to increase the network's coverage. The City of Amsterdam, where the movement started, was covered in 4 weeks and is currently served by more than 50 gateways. Local communities

around the world are encouraged to grow and deploy LoRa gateways, which can be purchased for a relatively small price from the organization's website.

2.4 European Regulations

In order to understand some of the limits that LoRa networks have to respect, it is necessary to look at how the "free" bands that these networks use are regulated by various entities. In fact, the fact that a band is unlicensed does not mean that it is also not regulated. There are essentially three levels at which different organizations handle spectrum allocation and limit its use:

1. At the *national level*, the frequency spectrum is managed by National Administrations, which:
    - Compile a table of spectrum allocations.
    - Define a framework for use of these bands.
    - Assign each band to different users, possibly via licenses.
2. At the *European level*, there are three organisms that cooperate to regulate spectrum usage:
    - The European Commission (EC)
    - CEPT's Electronic Communications Committee (ECC)
    - The European Telecommunications Standard Institute (ETSI)
3. At the *worldwide level*, the International Telecommunication Union (ITU) coordinates regional and national organisms.

For our purposes, we will from now on refer to the ETSI and ECC regulations. In particular, the documents of interest are [20] and [21].

LoRaWANs operate on Industrial, Scientific, and Medical (ISM) bands, in particular, in the $902 - 928$MHz band in the US and in the $863 - 870$MHz band in Europe [11]. These license-exempt bands are subject to regulations on radio emissions [20], thus radios are required to either adopt a Listen-Before-Talk (LBT)

policy or to duty cycled transmission, in order to limit the rate at which the end devices can actually generate messages. The latter policy is adopted in the vast majority of the cases. Furthermore, each transmission in a sub-band in the $863 - 870$MHz frequency range must respect some limitations on the emitted power. The following sections explore each one of these limits, and finally list a channel lineup for LoRaWANs, specifying the limits that LoRa devices must respect in every channel.

2.4.1 Effective Radiated Power (ERP) Limitations

There are mainly three ways to describe the emissions radiated by a device:

1. The electrical field strength (E) at a specified distance from the transmitting antenna.

2. The Effective Isotropic Radiated Power (EIRP), defined as the power that would have to be used on an isotropic antenna in order to get the same field strength that the tested device produces at the same distance. Conversion between field strength and EIRP is obtained using the following formula:

$$\mathrm{EIRP}_{\mathrm{dBm}} = 10 \log \left(\frac{E^2 \cdot r^2}{0.03} \right) \tag{2.8}$$

3. The ERP, defined similarly to the EIRP but using a half-wave dipole instead of an isotropic antenna. Since the gain of a half-wave dipole is at most 2.15 dBi (dB relative to an isotropic radiator), the following holds:

$$\mathrm{ERP} = \mathrm{EIRP} - 2.15 \tag{2.9}$$

ETSI uses the ERP metric to evaluate the emissions of a device. The way that power must be measured is described in two different clauses, based on whether the device under test includes an antenna or not. In the case the device is provided without an antenna, the vendor is also required to state the maximum gain of an antenna that can be connected to the device. In this case, the power must be measured at the connector, and the antenna gain needs to be factored in. In the case the device is shipped with an integrated antenna, the effective radiated

Table 2.4: Maximum on and minimum off times based on duty cycle definition, as specified in [21].

Duty cycle	Max "on" time / hour	Max "on" time	Minimum "off" time
$\leq 1.0\%$	36	3.6	1.8
$\leq 10\%$	360	36	3.6

power is the power radiated in the direction of the maximum field strength, under specified conditions of measurement. [20] also describes how the test to measure the radiated power should be performed: it specifies the set up of the antenna and receiver, polarization, height and other factors intended to make sure the measurement is performed in the direction where the antenna is emitting the most power.

2.4.2 Duty Cycle Limitations

Duty cycle is defined as the ratio, expressed as a percentage, of the maximum transmitter "on" time over one hour, relative to a one hour period. Duty cycle limitations are given in [20], Clause 7.10. These limitations apply to every receiver, excluding those with LBT capabilities. Since LoRaWAN has defined no LBT mechanism as of now, these limits must be respected by all LoRa devices.

Table 2.4 contains a couple of examples taken from [21], illustrating how a duty cycle limitation translates to a certain maximum time on air and minimum waiting time between consecutive packet transmissions: for example, a device with a 1% duty cycle can perform 10 transmissions of 3.6 seconds within one hour, while a device with a duty cycle of 10% can perform 10 transmissions of 36 seconds within one hour.

2.4.3 Channel Lineup

Three different ISM sub-bands can be distinguished in Europe [21], in the range specified by [11]:

1. The g1 line 3 ($867 - 868$MHz), g1.1 ($868 - 868.6$MHz) and g1.4 ($869.7 - 870$MHz) sub-bands, with maximum 36 seconds per hour Time on Air

Table 2.5: Channel lineup for LoRa according to ETSI regulations.

#	f	B	% of time on air	Max ERP	Regime
1	868.1	125 kHz	1%	14 dBm	g1.1
2	868.3	125 kHz	1%	14 dBm	g1.1
3	868.5	125 kHz	1%	14 dBm	g1.1
4	868.85	125 kHz	0.1%	14 dBm	g1.2
5	869.05	125 kHz	0.1%	14 dBm	g1.2
6	869.525	125 kHz	10%	27 dBm	g1.3

(ToA), 1% duty cycle to be shared between its sub-channels and a ERP limit of 14 dBm;

2. The g1.2 ($868.7 - 869.2$MHz) sub-band, with a maximum 3.6 seconds per hour ToA and 0.1% duty cycle to be shared between its sub-channels, ERP limit of 14 dBm;

3. The g1.3 sub-band: ($869.4 - 869.65$ MHz) with maximum 360 seconds per hour ToA and 10% duty cycle, ERP limit of 27 dBm.

Table 2.5 proposes an example lineup of 6 LoRa channels according to ETSI constraints on European ISM bands, taken from [22]. It is important to remark that, as long as the regulations in each frequency band are respected, an end device is allowed to transmit on different channels, contained in different sub-bands, in order to increase the aggregate ToA while still respecting the duty cycle limit in each sub-band.

3

Modeling a LoRa Network

A series of assumptions and simplifications must be made in order make the simulation of a LoRa network computationally viable. As for other system-level simulation tools, like the Vienna Long Term Evolution (LTE) simulator [23], the system-level simulations presented in this thesis are based on two components that aim at representing the actual transmission chain in a simplified way:

- A *link measurement model* is used to abstract the effects of propagation on signal strength, as well as to average out small scale fading and similar effects;

- A *link performance model* determines the probability of correctly receiving a packet at a reduced complexity, using only the information about link strength, interferers and other system-level effects.

The next section of this chapter is devoted to exploring how LoRa networks have been investigated and modeled in the scientific literature so far. After this introduction on the current state of the art, the remainder of the chapter will focus on each one of the two models mentioned above, explaining their various components and justifying the assumptions they are based on.

3.1 Related Work

The literature related to the work presented in this thesis is quite limited, since the interest of the research community in the relatively new technologies of LPWANs only started growing lately. This section is structured to provide an overview of the most significant contributions given by various academic papers, and categorize them according to the specific topic they were centered on: either modulation and propagation performance of LoRa devices, simulation of a LoRa system or other kinds of contributions, like protocols.

3.1.1 Modulation and Propagation Analysis

In [15] an exhaustive technical analysis of the LoRa modulation system is provided, comparing it with other LPWAN technologies, like Sigfox's UNB modulation. Thanks to the new understanding of the modulation and its analysis, the authors are able to formulate a co-channel rejection matrix, containing the values of SNR that are required for a packet of SF i to survive an interferer of SF j that is overlapping the desired signal. This matrix is represented in Eq. (3.20). The paper also evaluates the effect of interference by pure tones: an analysis shows that these are expected not to be a problem if they are less than 5 dB above the desired signal for SF 7 and less than 19.5 dB for SF 12. Unfortunately, the extent to which a Sigfox UNB signal can be assimilated to a pure tone is not analyzed, so the viability of a coexistence between the two technologies is still an open question.

In [24], some field trials of LoRa end-nodes are carried out in a urban environment, in order to evaluate the decoding performance of LoRa receivers, more precisely of the Semtech SX1276 chip. A first test of the sensitivity highlights that the improvement in sensitivity corresponding to an increase in the used SF is not as high as specified in official documents like [25]. The authors, though, note that results may have been better had the gateway been placed outdoors instead of indoors, where heavy shadowing may have been involved and may have worsened the performance of the LoRa link. Another series of tests analyze the range of a LoRa device: with a gateway placed outside a window at the second floor of a building, and the end device transmitting at a power of 14 dBm, a range

of 2800 m with a successful packet delivery rate of above 90% was achieved using SF 12. The experiment also confirmed that the choice of spreading factor heavily influences transmission range. Additionally, the authors note that the gateway placement was not ideal, and that an higher positioning would probably lead to better a performance.

In [26], the authors perform various experiments in order to better understand which factors may cause a packet loss in a LoRa network. In a first experiment, a "perfect" separation between different SF is confirmed. However, it is not clear whether there exists a threshold in the SINR values under which two spreading factors cannot be considered orthogonal anymore, as suggested in [15]. One additional aspect the authors cover is the capture effect in LoRa transmissions: only one of two concurrent transmissions with the same SF can be demodulated if one packet arrives at the receiver with an even slightly higher power than the other one. Figure 3.1 shows the probability of a certain event happening for different offsets in the transmission of a "strong" signal with respect to a "weak" signal. The three plots refer to, respectively, these events: the probability of correct reception of the weak signal; the probability of correct reception of the strong signal; the probability of losing both packets. It can be seen that for very negative offsets, when the last part of the strong packet only marginally overlaps with the beginning of the weak packet, both packets are received correctly with an high probability. As the offset grows towards 0 (representing a complete overlap of the two packets), the correct reception of the stronger packet is certain, while the weak packet is shadowed and its reception consistently fails. This goes on up to when the two packets are completely overlapping, and even when three symbols of the weak packet are sent before the strong packet's first symbol. At this point, the receiver is no longer able to correctly decode the stronger packet, and either packet is consistently lost because of the interference created by the other signal. This effect goes on up to when the offset is so high that the packets, again, overlap only for a few symbols. At that point, correct reception of both signals is once again possible. The authors of [26] state that it is especially useful to know of this effect in the case when beacons have to be sent synchronously by multiple gateways: the packets must be sent by all devices within a three symbol window, so that at least one of the packets will be correctly received. The paper also states that this synchronization (between 768 μs and 98.3 ms depending

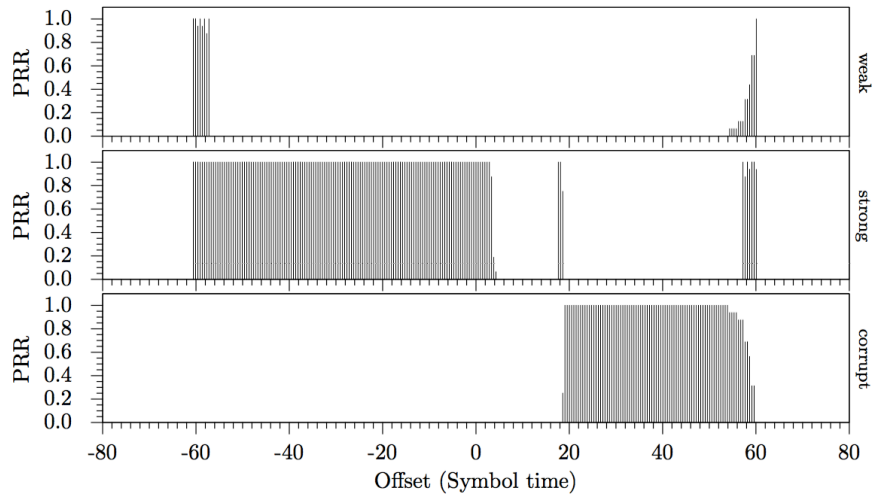


Figure 3.1: Capture Effect on LoRa devices.

on the SF and bandwidth settings) is easily achievable. Beyond the application suggested in [26], it should be noted that this effect may also be accounted for when modeling interference between two uplink packets concurring for detection by a gateway, in addition to what is described in [15], to achieve a more realistic representation of what happens at the link level.

Another contribution that [26] brings to the table is an evaluation of the Carrier Activity Detection (CAD) mechanism. This procedure is implemented in LoRa devices in order to detect the preamble of an incoming transmission. Precisely, this channel assessment requires to keep the radio on for a time of:

$$t_{\text{on}} = \frac{32}{B} + \frac{2^{\text{SF}}}{B}. \tag{3.1}$$

Some additional time is then needed for processing the received data:

$$t_{\text{proc}} = \frac{\text{SF} \cdot 2^{\text{SF}}}{1750 \cdot 10^3}. \tag{3.2}$$

This translates to the fact that CAD can be performed in approximately 2 symbol periods, and only requires the radio to be on for about 1 symbol period. A series of experiments state that CAD success rate is above 97%, with a false positive rate of 0.092%. Using different SF and bandwidth settings does not seem to affect the performance. It should be noted that false positive rates would

certainly increase in the case of co-existence of multiple LoRa networks, since devices belonging to a different LoRa network would use signals that are identical to those intended to be picked up by the CAD mechanism.

[27] and [28] focus on evaluating propagation of devices employing the LoRa modulation. In the measurements conducted in [27] the gateway antenna was placed at 24 m above sea level, and an end device employing the Semtech SX1272 chip, set up to periodically transmit data at a power of 14 dBm, was first mounted on a car and then on a boat to measure successful packet delivery rate. With a spreading factor of 12 and a bandwidth of 125 kHz, the gateway sensitivity was found to be of -137 dBm. Within a 2 km range from the base station, 12% of packets sent from the ground were lost. In the 2–5 km range, packet loss ratio stayed below 15%, while one third of the packets were lost in the 5–10 km range. Finally, 75% of packets transmitted from 10–15 km were lost. Range was significantly extended when the end device was placed on a boat: the LoRa modulation was able to achieve, albeit inconsistently, a range of 30 km. The same authors provided some more results for indoor propagation in [28], where a device placed in different rooms of a building, staying always within 300 m of the gateway, consistently delivered messages with a success probability above 94%. Additionally, the standard deviation of shadowing was estimated to be in the $[4.95 - 10.5]$ dB range, depending on the position inside the building.

3.1.2 Simulations

The same authors as [26] continued their work in [29] by developing a system-level simulator for a LoRa network, written using the python framework *SimPy*. This work leverages the discoveries made in [26], especially regarding the modeling of the capture effect. The assumptions of perfect orthogonality between different spreading factors are used in this paper, too. The main metric used in the simulation to assess the quality of the system is called DER, and is computed as the ratio of received messages to transmitted messages over a period of time. One of the result plots is reproduced in Figure 3.2. The three curves represent the DER for different settings in the SFs used by the nodes in the network. SN^3 refers to a setting where all nodes employ the same spreading factor, while SN^4 and SN^5 assign the SF to each node based on their position, trying to minimize

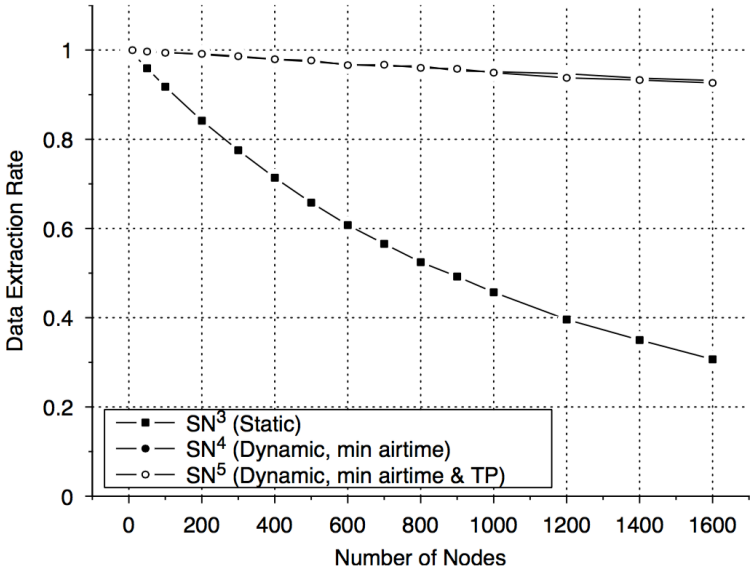


Figure 3.2: DER for a realistic distribution of SF. Result from [29].

airtime first and then, in the case of SN^5, setting the minimum transmit power that would still allow the node to get to the gateway successfully, in order to reduce the amount of interference. If the requirement for the rate of successful reception of a packet is set at DER $> 90\%$, the graph in Figure 3.2 shows that 1600 nodes can be supported by a single gateway. An additional experiment is performed in [29], trying to assess the effect of having multiple sinks serving a network. Unfortunately, this simulation round assumes that every node uses the same SF: this setting cannot be considered realistic (SF diversity is desirable because it increases channel capacity, and would be enforced by an ADR algorithm), and it yields skewed results since it does not allow the network to leverage the orthogonality of different spreading factors to decrease the number of collisions.

The main result brought by [24] shows that LoRaWAN achieves a throughput equal to that of ALOHA networks, since the protocol is essentially the same, however this simulation has some key differences with respect to the approach that was followed in this thesis. Spreading factors are considered to be perfectly orthogonal, differently from what was discovered in [15]. Additionally, as soon as two packets overlap in time the authors consider that as a collision, and both packets are marked as lost, whereas papers such as [15, 29] showed that this is not

always the case. Packets of uniformly distributed length are modeled to arrive according to a Poisson law, while more realistic periodic reporting models have been used in this thesis, and no propagation loss model was accounted for. No information is given regarding how the choice of which spreading factors to assign to which device was made.

[30] shows how duty cycle limitations can have a significant influence on throughput, by throttling the amount of sent packets and lowering the probability of collisions. Again, it's assumed that collisions happen as soon as two packets overlap in time. Additionally, the authors show that an application level payload of 10, 30 and 50 bytes does not critically affect the performance of the system.

3.1.3 Other Aspects of a LoRa Network

Another important contribution presented in [26] is the formulation of a multi-hop network protocol for LoRa devices, called LoRaBlink. The protocol is assumed to be employed in a low density network, with low traffic volume and a limited number of nodes, and leverages a beacon-like synchronization between devices to allow nodes to switch to sleep mode as frequently as possible, while still forwarding packets and achieving a tolerable latency in a slotted access fashion. The protocol was also tested in a real deployment, featuring six nodes and one sink. The network was able to deliver packets with a success rate of 80%, and it has been projected that each node would have a life of 2 years on 2 AA batteries. Such a protocol would be especially useful in very sparse networks, where gateways would not be easy to install: relying on message forwarding devices would allow a LoRa network to cover an area that is greatly vaster than it is currently possible with LR-WPANs that exploit multi-hop, thanks to the increased range of LoRa messages.

3.2 Link Measurement Model

Given a transmitter-receiver pair, the link measurement model aims at estimating the strength of the signal at the receiver side. This model needs to account for various factors, like transmit power, shadowing, fast fading and antenna gains, both at the receiver and at the transmitter.

Let us denote the transmitter and receiver antenna gain with G_{tx} and G_{rc}, respectively, and the transmit power with P_{tx}. Then, the received power can be expressed as:

$$P_{\text{rx}} = \frac{P_{\text{tx}} G_{\text{tx}} G_{\text{rc}}}{L} e^{\xi}\,, \tag{3.3}$$

where L is the path loss and e^{ξ} is a factor modeling the slow fading effect, also called shadowing, via a lognormal random variable with $\xi \sim \mathcal{N}(0, \sigma^2)$. We assume that the fast fading Rayleigh-distributed component is already averaged in the link performance model.

In the logarithmic domain, Eq. (3.3) becomes

$$\begin{aligned} P_{\text{rx}}^{\text{dB}} &= P_{\text{tx}}^{\text{dB}} + G_{\text{tx}}^{\text{dB}} + G_{\text{rc}}^{\text{dB}} - L^{\text{dB}} + 10\xi \log_{10} e \\ &= P_{\text{tx}}^{\text{dB}} + G_{\text{tx}}^{\text{dB}} + G_{\text{rc}}^{\text{dB}} - L^{\text{dB}} + 4.34\xi\,. \end{aligned} \tag{3.4}$$

Path loss L^{dB} consists in the sum of two contributions: the *propagation loss*, which depends on the distance between transmitter and receiver, and the *building penetration loss*, due to the wall attenuation:

$$L^{\text{dB}} = L_{\text{propagation}}^{\text{dB}} + L_{\text{buildings}}^{\text{dB}}\,. \tag{3.5}$$

3.2.1 Propagation Loss Model

According to [31], the propagation loss (also called external path loss) can be computed as:

$$\begin{aligned} L_{\text{propagation}}^{\text{dB}} = 40(1 - 4 \times 10^{-3} \times h) \log_{10} R|_{\text{km}} \\ - 18 \log_{10} h|_{\text{m}} + 21 \log_{10} f|_{\text{MHz}} + 80\,, \end{aligned} \tag{3.6}$$

where $h \in [0, 50]$ m is the gateway antenna elevation, measured from the average rooftop level. We want to remark that the antenna elevation has a massive impact in the performance of the system [32]. Assuming $f = 868$ MHz and $h = 15$ m, it follows that:

$$L_{\text{propagation}}^{\text{dB}} = 120.5 + 37.6 \log_{10}(R|_{\text{km}}) \; = I^{\text{dB}} + 37.6 \log_{10}(R|_{\text{km}})\,. \tag{3.7}$$

This yields:

$$\begin{aligned} L_{\text{path}}^{\text{dB}} &= I^{\text{dB}} + 37.6 \log_{10} R|_{\text{km}} \\ &= 10 \cdot 12.05 + 10 \cdot 3.76 \log_{10} R|_{\text{km}} \\ &= 10 \log_{10}(1.12 \cdot 10^{12}) + 10 \log_{10} R|_{\text{km}}^{3.76} \\ &= 10 \log_{10} I + 10 \log_{10} R|_{\text{km}}^{3.76} \\ &= 10 \log_{10} IR|_{\text{km}}^{3.76}, \end{aligned} \tag{3.8}$$

where R is measured in km.

3.2.2 Building Penetration Loss

In order to model the losses that are caused by external and internal walls of buildings, we resort to the model described in [3]. The overall building penetration loss $L_{\text{buildings}}^{\text{dB}}$ is the sum of the following three contributions:

1. losses caused by the external walls of buildings, called EWL;
2. losses caused by the internal walls of buildings;
3. gain in received power thanks to the fact that a device is above the first floor.

EWL for a device is modeled as a uniform random variable that takes values in a certain range: EWL $\sim \mathcal{U}(r)$. The three possible ranges and the probability that a node experiences that kind of loss are listed in Table 3.1. The fact that two devices will not necessarily experience the same external wall penetration loss is intended to model the variety of materials and thickness of external walls in a wide variety of different buildings.

Table 3.1: Possible distributions of the EWL uniform random variable.

Probability	Range r
0.25	$[4, 11]$ dB
0.65	$[11, 19]$ dB
0.1	$[19, 23]$ dB

The contribution of internal walls is expressed as the maximum value between of two values. The first one represents the loss due to the number of internal walls:

$$\text{Tor1} = W_i \cdot p\,, \tag{3.9}$$

where W_i is uniformly distributed in the $[4, 10]$ dB range, and p represents the number of internal walls separating transmitter from receiver. It is assumed that $p = 3$ for 15% of the devices, and that the rest of the devices are equally distributed among the values of $p = \{0, 1, 2\}$. The second value that is needed to model internal wall loss is:

$$\text{Tor3} = \alpha d\,, \tag{3.10}$$

where $\alpha = 0.6\,\text{dB/m}$ is the penetration distance coefficient, and d is uniformly distributed in the $[0, 15]$ m range.

Finally, the GFH contribution accounts for the better reception that an increase in height yields:

$$\text{GFH} = nG_n\,, \tag{3.11}$$

where $G_n = 1.5\,\text{dB/floor}$ is the gain due to an increase in height caused by a single floor and n, representing the number of floors, is uniformly distributed among the values of $n = \{0, 1, 2, 3, 4\}$.

Combining these three contributions, the total loss due to buildings for an indoor end device is:

$$L^{\text{dB}}_{\text{buildings}} = \text{EWL} + \max(\text{Tor1}, \text{Tor3}) - \text{GFH}\,. \tag{3.12}$$

3.2.3 Correlated Shadowing

Many studies on shadowing in wireless networks can be found in the literature. In particular, [33] provides a structured synthesis of the existing literature on the modeling of correlation in wireless shadowing: research suggests that shadowing correlation significantly affects various system-level phenomena, like handover behavior, interference power and performance of macrodiversity schemes. Because of this, an appropriate model for the correlation must be considered in system-level simulations. Two kinds of correlation are usually considered [34]:

1. If a transmitter sends a message to a receiver, we expect the amount of

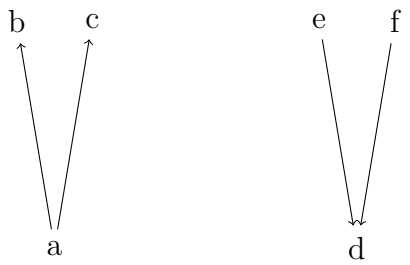


Figure 3.3: Illustration of the two kinds of shadowing correlations.

shadowing experienced by the receiver to be correlated with the shadowing affecting any other device that is "close" to it. This correlation is a function of the distance separating the two devices, and is usually modeled with an exponential function [35]. This case is illustrated by nodes a, b and c in Figure 3.3: since b and c are close in space, their line of sight will likely be blocked by the same kinds of obstacles, and the shadowing at those two positions will hence be correlated.

2. If two devices which are close to each other transmit a signal, we expect the receiver to receive signals that are affected by two correlated shadowing values. This effect is described as site-to-site cross-correlation in [34]. This case is illustrated by nodes d, e and f in Figure 3.3: since two close nodes are communicating with the same point, we expect that if d is blocked by a big object from e's perspective, it will be blocked from f's perspective, too.

The most common correlation model is a decaying exponential of distance, [33, Sec. VI-B]. Denoting the distance between end-nodes i and j with $d_{i,j}$, the shadowing correlation is

$$\rho_{i,j}(d_{i,j}) = e^{-d_{i,j}/d_0}\,, \tag{3.13}$$

where $d_0 > 0$ is a tunable parameter, called decorrelation distance: this represents the distance at which correlation between two shadowing random values is under the e^{-1} threshold, and the values are considered to be reasonably uncorrelated.

As for the implementation of correlated shadowing components, the most common way in the literature involves the generation of shadowing maps, or 2D functions that describe shadowing at each point in the map for a given transmitter

position. One of the conventional ways to generate such maps exploits Cholesky factorization [33], which is however computationally expensive especially when the number of points that are needed grows considerably, as happens in our simulations. In order to reduce the computational effort required to produce the maps with respect to Cholesky decomposition methods, [36] proposes an alternative method to gradually build a map by correlating new samples with close ones that were already generated. Another approach used in [34] involves filtering a map of independent Gaussian random variables to get a spatially correlated process.

However, in order to simulate a urban scenario with tens of thousand of nodes, as envisioned for a LoRa network, this thesis resorts to an heuristic approach validated by [37]. Assuming a shadowing decorrelation distance $d_0 = 110$ m [3], we generate a regular grid in which each square has a side length of d_0 and draw an independent Gaussian random variable at each vertex of the grid. To calculate the shadowing values of nodes that are not exactly placed on a vertex of the grid, we need to interpolate the values of the grid. We can do this while still respecting the correlation between data points in the grid if we use the appropriate interpolation coefficients, as explained in [37].

Let's define a set S of positions in space, and a corresponding subset $s_i \in S\,, i \in \{1, \ldots, N\}$. Assume we are given a set of observations X_i of a random variable $X \sim \mathcal{N}(0, \sigma^2)$, one in correspondence to each position s_i. In our case, the X_i values are randomly generated shadowing values at each point of the grid. Let's now define the covariance function between two points s and s':

$$k(s, s') = \rho(d_{s,s'}) = e^{-d_{s,s'}/d_0}\,. \tag{3.14}$$

Given the function above, we can now compute the covariance matrix:

$$K = \begin{pmatrix} k(s_1, s_1) & \cdots & k(s_1, s_N) \\ \vdots & \ddots & \vdots \\ k(s_N, s_1) & \cdots & k(s_N, s_N) \end{pmatrix} \tag{3.15}$$

where $k(s_i, s_j)$ is the covariance between the datapoints at the i-th and j-th position. Note that we can compute this function between any couple of points, regardless whether we have data for that position or not, since the covariance

function only depends on their reciprocal distance.

Now, we can define a set of *weight functions* $\phi_1, \ldots, \phi_N$ such that the following two properties are satisfied:

$$\phi_i(s_j) = \delta_{ij} \qquad \forall i, j = 1, \ldots, N \tag{3.16}$$

$$\sum_{i=1}^{N} \phi_i(s) = 1 \qquad \forall s \in S. \tag{3.17}$$

In order to find the appropriate weight functions to interpolate the X_i values, such that we obtain a value for each point in S that respects the autocorrelation function k, we can use the following formula:

$$\phi_i(s) = \sum_{j=1}^{N} K_{ij}^{-1} \cdot k(s_j, s). \tag{3.18}$$

Note that the K^{-1} matrix in the equation above does not change with s, so it can be computed once and used for the computation of every ϕ coefficient. Interpolated values can finally be obtained as:

$$X(s) = \sum_{i=1}^{N} X_i \cdot \phi_i(s). \tag{3.19}$$

The procedure described above captures correctly the first one of the two aspects of the shadowing correlation: given a transmitter, the shadowing experienced by two receivers that are close to each other will be correlated. In order to also express the fact that a receiver "sees" correlated shadowing values on transmissions by devices that are close to each other, in the model we make use of the same shadowing map for every point belonging to the same square in the grid. This way, if two reasonably close nodes are transmitting towards the same base station, the amount of shadowing the gateway experiences is the same for both signals.

Table 3.2: Sensitivities of Gateways and End Devices to different Spreading Factors.

SF	$S_g(i)$	$S_e(i)$
7	-130.0	-127.0
8	-132.5	-129.5
9	-135.0	-132.0
10	-137.5	-134.5
11	-140.0	-137.0
12	-142.5	-139.5

3.3 Link Performance Model

Given the receive power of transmissions in the system, which are computed thanks to the Link Measurement Model, and a knowledge of where and when said transmissions originated, the link performance model aims at abstracting the real implementation of the physical layer receive chain and at making interference computations more manageable. More precisely, the model abstracts the performance of the Semtech SX1301 chip that is used in gateways and of the Semtech SX1272 chip normally used in end devices, emulating their parallel decoding, sensitivity and interference resistance performances.

3.3.1 Receiver Sensitivity

Let us denote with $S_g(i)$ and $S_e(i)$, respectively, the sensitivities in dB of the gateway and end device receivers for $SF = i$. These sensitivities are summarized in Table 3.2 [10].

For each value in Table 3.2, we also need to factor in the gain of the receiver antenna G_{rc}, improving the reception by improving the sensitivity value for $G_{\text{rc}} > 0$. It can be seen that an increase of SF yields a better sensitivity, with regular steps of 2.5 dB. In case of Downlink (DL) transmissions, since the sensitivity of an end device is assumed to be worse than the sensitivity of a gateway, we introduce an offset of 3 dB.

Sensitivity values are used in order to determine whether the packet is detected by a device or not: as an example, any signal with $SF = i$ whose power at the

receiver location is below the threshold $S_g(i)$ cannot be detected by the gateway; if, instead, the received power is above the required sensitivity, then it can be detected. In this case, we also assume that the receiver will lock on the incoming signal and start receiving the packet.

A further assumption regards the received power of the packet, which is computed thanks to Eq. (3.4) and assumed to be constant for the whole duration of the reception. This implies that when a packet is received with a high enough power to start being detected, it will be detectable (i.e., above the sensitivity) for the rest of the time it takes to be completely received. Finally, it's also assumed that if two or more signals, whose individual powers are all below sensitivity, are simultaneously arriving on the receiver antenna, they cannot be detected by the receiver even if the sum of their power gives a value that is above sensitivity. This assumption is justified by the fact that, even if the receiver started reception of one of the packets because it sensed that the channel was busy, the signal would be destroyed by interference from the other signals that contributed to raise the reception power above the sensitivity threshold.

3.3.2 Interference

Since our objective is to simulate the behaviour of a standalone LoRaWAN network, we assume that interference can only come from other LoRa transmissions. By making this assumption, we can leverage the partial orthogonality property of different SFs to model whether a packet survives interference from other LoRa transmissions or not. Let us introduce the following (relative) SINR threshold matrix [15]:

$$\mathbf{T} = \begin{bmatrix} 6 & -16 & -18 & -19 & -19 & -20 \\ -24 & 6 & -20 & -22 & -22 & -22 \\ -27 & -27 & 6 & -23 & -25 & -25 \\ -30 & -30 & -30 & 6 & -26 & -28 \\ -33 & -33 & -33 & -33 & 6 & -29 \\ -36 & -36 & -36 & -36 & -36 & 6 \end{bmatrix} . \tag{3.20}$$

The element $T_{i,j}$ in the matrix above is the SINR margin (in dB units) that a packet sent with $SF = i$ must have in order to be correctly decoded if the

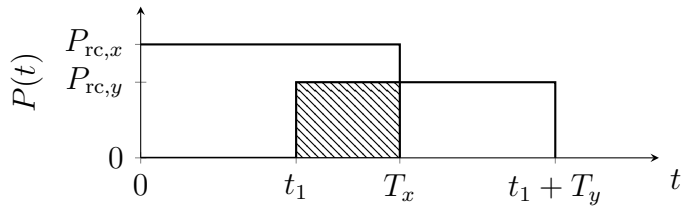


Figure 3.4: Power *equalization* of colliding packets. The highlighted energy is spread on the duration of the packet.

interfering packet has $SF = j$. We remark that, in presence of multiple interfering packets, we need to satisfy the margin conditions with all the interfering packets, summing the received power values for each SF.

Therefore, referring to the Single Input Single Output (SISO) case [23, Sec. III-C3], we recall the general definition of SINR:

$$\mathrm{SINR} = \frac{P_{\mathrm{rc},0}}{\sigma_w^2 + \sum_{l=1}^{N_{\mathrm{int}}} P_{\mathrm{rc},l}}\,, \tag{3.21}$$

where $P_{\mathrm{rc},0}$ is the power of the packet under consideration, N_{int} is the number of interfering packets, and $P_{\mathrm{rc},l}$ is the power of the l-th interfering packet. Focusing on an end device using $SF = i$ and a set $\mathcal{I}_j$ of interferers using $SF = j$, we define

$$\mathrm{SINR}_{i,j} = \frac{P_{\mathrm{rc},0}}{\sigma_w^2 + \sum_{l\in\mathcal{I}_j} P_{\mathrm{rc},l}}\,. \tag{3.22}$$

Therefore, a packet with $SF = i$ is correctly decoded if, for every j (i.e., for every set $\mathcal{I}_j$ of interfering packets with the same SF), the following inequality holds:

$$\mathrm{SINR}_{i,j}^{\mathrm{dB}} > T_{i,j}\,. \tag{3.23}$$

A further remark must be made. The elements in matrix $\mathbf{T}$ are calculated assuming that the two packets are perfectly overlapping. However, in the general case, packets are not perfectly synchronized. Because of this, we must *equalize* the interfering power value for the computation of the SINR. Consider the situation illustrated in Figure 3.4, in which a packet with $SF = x$ is received at time $t = 0$ and whose transmission lasts T_x. A packet with $SF = y$ is received at time $t = t_1$ and its transmission lasts T_y. The energy of packet x is $E_x = P_{\mathrm{rc},x}T_x$, while the

interfering energy is $E_y^{\text{interf}} = P_{\text{rc},y}(T_x - t_1)$. Therefore, the *equalized* interfering power is:

$$P_{\text{rc},y}^{\text{interf}} = \frac{E_{\text{rc},y}^{\text{interf}}}{T_x} = \frac{P_{\text{rc},y}(T_x - t_1)}{T_x} = P_{\text{rc},y}\left(1 - \frac{t_1}{T_x}\right). \tag{3.24}$$

Similarly to the example above, we assume that in general the interfering energy for any reciprocal position of the signal and an interferer can be "spread out" on the signal in order to then compute the SINR using Eq. (3.22). Denoting with t_{ol} the period of time during which the interferer is overlapping with the useful signal, the general formula becomes:

$$P_{\text{rc},y}^{\text{interf}} = \frac{P_{\text{rc},y} \cdot t_{\text{ol}}}{T_x}. \tag{3.25}$$

This assumption is justified by the fact that the underlying channel code employed by the modulation, as remarked in Section 2.2, makes use of an interleaver: even if the interference is concentrated on a few consecutive symbols, we can assume that a good interleaver will spread it out and allow the channel code to eventually correct the errors caused by the interferer.

Moreover, thanks to the channel coding technique used by the LoRa modulation standard, we also assume that we will always correctly receive a packet that is above sensitivity and survived interference, due to the fact that the curves of the bit error rate versus SINR decline very sharply as SINR grows above the thresholds reported in matrix $\mathbf{T}$ in Eq. (3.20).

Eq. (3.20) allows us to make some interesting observations about the behavior of interference between LoRa packets. First of all, we notice that one of two signals employing the same spreading factors and having similar receive power can both be correctly decoded, if they overlap in time for a sufficiently small amount of time so that the SINR of each one with respect to the other after equalization as per Eq. 3.25 is above 6 dB. This models the same effect that was observed in [26], as shown in Figure 3.1 for very high and low offsets. Eq. (3.20) also states that some spreading factors are more resistant to interference than others: a transmission employing SF 7 can survive interference by a signal with SF 12 that is up to 20 dB stronger, however packets with SF 12 are much more resilient, since they are still able to be decoded correctly if an interferer of SF

7 is up to 36 dB stronger. The general trend that can be observed is that an increase in spreading factor brings 3 dBs of added resistance to interferers. This asymmetry leaves room for considerations on the best distribution of SFs in a LoRaWAN network, balancing range, interference and throughput.

3.3.3 Gateway Model

We assume that a single LoRa gateway is capable of emulating 8 receivers working in parallel, as explained in [10]. These 8 *receive paths*, as depicted in Figure 2.3, are connected to the same antenna, and are assumed to have the following characteristics:

- The center frequency of each receive path must be individually configured. Each receive paths can be centered on a different frequency, and multiple receive paths can be centered on the same frequency.
- Any SF can be received without prior configuration on any receive path.
- When more than one receive path is listening to the same channel, we assume that they can manage in parallel as many packets as the number of listening receive paths. The packets may even have the same SF. In other words, if there are multiple receive paths on the same frequency and a packet arrives, only one receive path "locks" on the incoming signal, leaving the other ones free to sense more incoming packets, and possibly allow the correct detection of multiple overlapping signals.
- If a packet arrives at a certain LoRa channel and there are no available receive paths listening at that channel, the packet is lost.

3.4 Other Models

3.4.1 Channel Access

In the following of this thesis we explicitly refer to a LoRa Class A network [6], where transmissions are always initiated by the end devices. For this purpose, the end-node may choose at random one channel in the available channel lineup,

which is shared between all LoRa devices connected to the network. Unless stated otherwise, this thesis will use the basic channel lineup mandated by [11] in Europe. Since the channels will be chosen at random, the available gateway receive paths are equally distributed among the three channels as shown in Table 3.3 in order to maximize the coverage in each channel.

Table 3.3: Channel lineup used in the simulations.

Channel index	Frequency	Number of receive paths
1	868.1	3
2	868.3	3
3	868.5	2

3.4.2 Application Model

Devices are modeled to generate traffic according to the Mobile Autonomous Reporting (MAR) periodic reports model described in [3, Sec. E.2.2]. Periodic uplink reporting is expected to be a common behavior in the IoT scenario, where multiple devices will have to monitor the conditions of utilities like gas, water and electricity and of the environment, reporting periodically measures of temperature and humidity. This model is particularly useful for capacity evaluations.

According to the MAR periodic reports model, the application payload size is a Pareto-distributed random variable, with shape parameter $\alpha = 2.5$ and minimum application size payload of 20 bytes and a cutoff of 200 bytes: this means that packets with a payload greater than 200 bytes are trimmed to a size of 200 bytes. The distribution of devices with a certain packet inter-arrival time is described in Table 3.4.

In this work, no DL transmissions (messages from the gateways to the end devices) are considered. This, however, should not be considered an heavy limitation, since most of the traffic in LoRaWAN networks is expected to be Uplink (UL).

In order to adapt this distribution to the LoRa specification, it was changed to have a minimum payload size of 10 bytes and a cutoff of 50 bytes. Furthermore,

Table 3.4: Distribution of packet interarrival times.

Packet interarrival time τ	Percentage of devices
1 day	40%
2 hours	40%
1 hour	15%
30 minutes	5%

every end device is assigned a random initial reporting delay, after which the node generates a new packet every τ seconds, in order to avoid all devices to be synchronized.

From the distribution it can be seen that almost half of the devices are expected to have a very small packet arrival rate of 1 packet/day. On the other hand, 5% of devices will each generate 48 packets every day: this means that the aggregate traffic of a small number of frequently transmitting devices is expected to be six times the traffic generated by 40% of the infrequently transmitting nodes. This figure gives an idea of the impact that periodicity settings in an even small amount of devices can have on the performance of the whole network.

4

Simulation of a LoRa network

To perform the network simulations of a LoRa system the Network Simulator 3 (NS3) software [4], an open source Discrete Event Simulation (DES) suite, has been used. The simulator has been expanded with the creation of a `lora` module that implements the various models described in Chapter 3. This chapter first provides a brief introduction to the NS3 software and then describes the structure and implementation of the new `lora` module.

4.1 Network Simulator 3

NS3 is a network simulation software intended for research and educational use, licensed under the GNU General Public License (GPL) and developed by a community of users. By combining several C++ objects, with each class modeling an aspect of a network, NS3 is able to simulate complex networks in a detailed and realistic fashion. Classes that model related concepts or systems are grouped in modules: as an example, the `wifi` module contains several classes that model components of a WiFi system, such as Access Points, WiFi enabled devices, the WiFi MAC layer and a wireless channel. These classes are interconnected and, when combined with some other modules that model core functionality, device mobility, propagation and so on, they can be used to simulate a whole network

implementing the WiFi standard.

The fact that NS3 works according to the DES principle means that a simulation consists of a series of *events*, each one tied to a certain time. The simulator's task is to execute these events (i.e., perform the appropriate function call that is linked to that event), which result in a change of the state of the simulation and possibly in the scheduling of more events. An example of event is the transmission of a packet on a wireless channel: this is represented by a function call from the class representing the device's PHY layer to the one representing the channel, which in turn will schedule an event representing reception of the packet by another device's PHY layer, possibly after a channel delay. Once an event is executed, the simulator moves on in the events list to perform the next function call. Since one event may schedule multiple other events, it is not guaranteed that at some point the simulation will end. In such cases, a special stop event can be issued to terminate the simulation. NS3's event driven approach ensures that, even if a simulation features only two events that are scheduled far away in time one from the other, the simulator will execute them one directly after the other. This approach speeds up simulations while still keeping them realistic, since no change in the state of the system was scheduled to happen between the two events.

NS3 also provides a built-in Pseudo Random Number Generator (PRNG), namely the MRG32k3a by Pierre L'Ecuyer, described in [38]. The generator provides $1.8 \cdot 10^{19}$ independent streams of random numbers, each one consisting in $2.3 \cdot 10^{15}$ substreams. Each substream has a period of $7.6 \cdot 10^{22}$: this means that the period of the generator amounts to $3.1 \cdot 10^{57}$. Random variables are provided as objects that access one of the independent streams of the underlying random number generator, and output random numbers according to a certain distribution described by a set of parameters. The fact that each random variable is assigned a different PRNG stream ensures that there will be no correlation between different random variables. A system to perform different "runs" of a simulation, ensuring that each repetition makes use of a distinct stream, is also available.

NS3 also features a tracing system, used to monitor variables during the simulation, and optionally trigger an action when a change is detected. This system is used to gather data during simulations on optimized runs, when logging is

disabled and no data can be gathered through the program's standard output.

While the models that are specific to a network or protocol are implemented in a module's classes, the topology and models that are to be used in a specific NS3 simulation are described via a C++ or Python program, which typically follows the following steps:

1. *Creation of a topology*: the set of nodes (i.e., devices) that will be used in the network is instantiated as a collection of `Node` objects. A `MobilityModel` may be associated to each node, to represent the node's physical position and how it changes with time.

2. *Models*: a certain protocol stack is installed on the previously created set of nodes. This is usually done via *helpers*, classes that are specialized in installing on a node the various objects implementing the needed layers of the ISO/OSI stack. This step gives each node the ability to create, send, receive and interpret packets belonging to that protocol.

3. *Configuration*: the models of a protocol are configured to use certain values as their parameters, and links between different nodes are created. Usually, this is done by "subscribing" multiple nodes' PHY layers to the same channel object.

4. *Execution*: the simulation is started, and the `Simulator` class goes through the events and executes the corresponding function calls. During the simulation, trace sources fire and save data either in appropriate data structures or on a file. In some cases, it can be useful to stop a simulation (i.e., schedule a stop event) once enough data has been collected from trace sources.

5. *Performance analysis*: after the simulation is stopped the gathered data can be analyzed and visualized.

The next section will focus on explaining how the models for a LoRa network have been created and describes the various ways they can be configured.

4.2 The `lora` Module

In order to model the behavior of a LoRaWAN, a new `lora` module was created. This module is essentially a collection of classes that work together to describe the behavior of LoRa EDs and GWs at various levels, from the PHY to the Application layer. The set of classes that are needed to simulate the protocol stack on a device can be seen in Figure 4.1. Additionally to those classes representing a layer in the stack (`LoraPhy` and `LoraMac`), some other classes were used to model aspects of the system like losses caused by buildings, correlated shadowing, interference and duty cycle limitations. The structure and interactions of the code will be described briefly in the following of this section, starting from the upper layers and ending with the class representing the wireless channel.

The work described in this thesis is focused on the ED and GW implementations, since it's at this level that the LoRa modulation is employed. Creation of the NS was not considered a priority, since investigation of the effects of ADR and MAC commands on the network were not part of this thesis' objectives.

4.2.1 `PeriodicSender`

The application layer class `PeriodicSender` consists in a packet generator which creates zero-filled packets of a randomized payload size, according to Section 3.4.2. It should be noted that creating more "realistic" payload contents would not affect the simulation results, since our link models don't account for the contents of the packets in the link abstraction.

The application transmission period `m_interval` determines the delay of the "transmit a random packet" event that is scheduled right after a transmission, and can be set up as an attribute of the class. Note that, at the application level, transmission simply means that the packet is forwarded down to the LoRa MAC layer. Since the function call that transmits a packet also schedules the next transmission, this application will keep sending packets until it is stopped via a specific function call. When the application is first started on a node, a random delay for the first packet sending event is chosen via a random variable $d \sim \mathcal{U}([0, \texttt{m\_interval}])$.

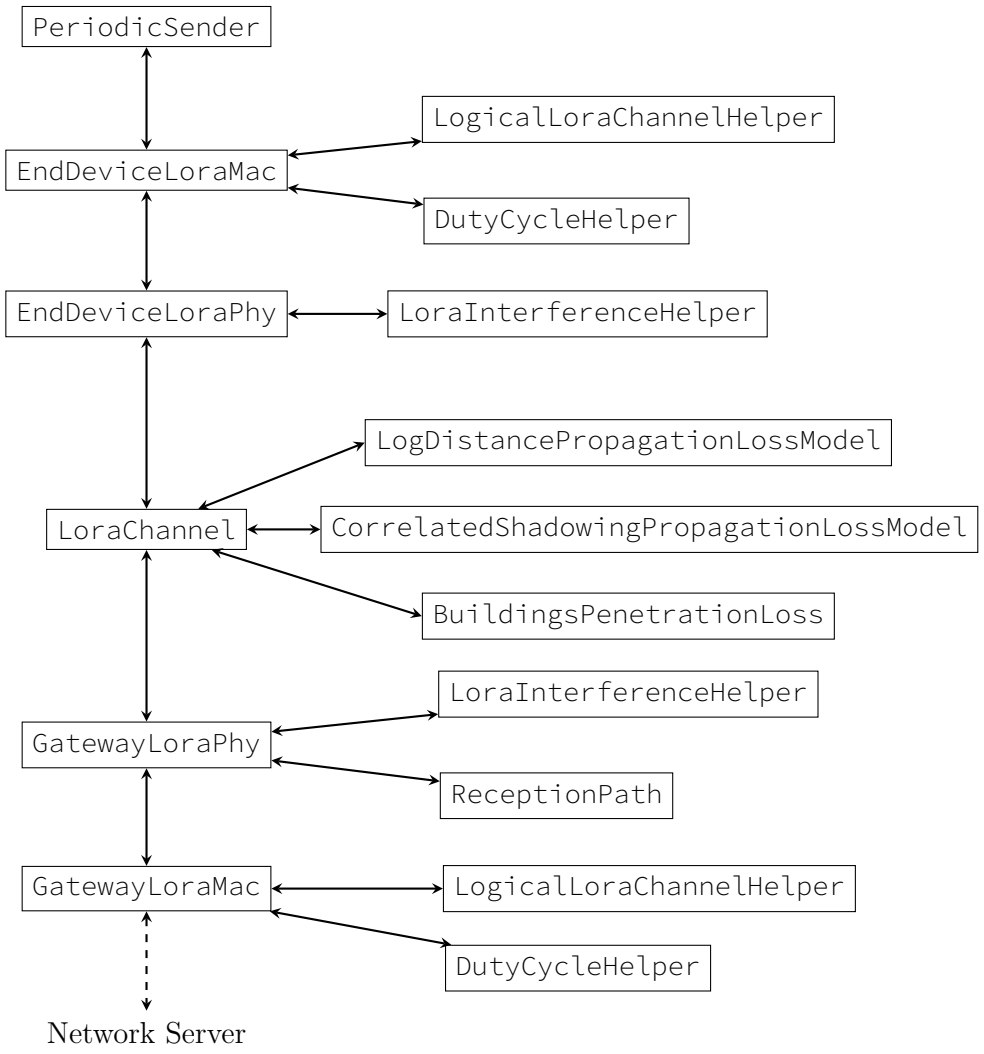


Figure 4.1: The LoRaWAN stack as it was represented in the `lora` module.

4.2.2 LoraMac

The LoraMac class models the MAC layer of a LoRaWAN device. This class is used to keep track of the available network channels via a LogicalLoraChannelHelper object, and to account for the duty cycle limitations demanded by the regulations: it's this class's responsibility not to send messages coming from the upper layer if this would mean breaking the duty cycle rule, and to queue them and send them at a more appropriate time. In order to keep track of different duty cycles on different sub-bands and to separate the duty cycle logic from the class, a DutyCycleHelper class was created. More specifically, the two subclasses EndDeviceLoraMac and GatewayLoraMac implement behaviors that are specific to an ED and a GW, respectively.

EndDeviceLoraMac is the object where a device's class is defined and influences the behavior of the ED: since this class controls the state of the underlying PHY layer, it is also required to correctly handle waking up the radio from sleep when a receive window needs to be opened, or to continuously listen when a Class C needs to be implemented. While the current implementation only supports Class A devices, implementation of other behaviors in future iterations of the work will be straightforward, as it will leverage inheritance to implement specific behaviors in subclasses. Algorithm 4.1 contains the procedure by which a packet is taken from the application layer by the MAC layer and passed to the underlying PHY layer, and shows how a random channel is picked by the ED to initiate a transmission. The procedure as illustrated in Algorithm 4.1 only keeps one packet in the queue at a time, however different implementations where a queue of backlogged packets is kept are still possible at the price of a slightly higher complexity.

The GatewayLoraMac class differs from EndDeviceLoraMac in that it implements a simpler, forward-only mac layer. These classes are also responsible for the interpretation of MAC commands that are either piggybacked in the FOpts field or contained in the FRMPayload, even though these functionalities are not implemented as of now. LoRa packets, with their specific structure, were implemented as extensions of the basic packet class.

Procedure 4.1 Send procedure in the `EndDeviceLoraMac` class.

Input: The `packet` variable contains the application-level payload

```
1: Add the Frame and Mac headers to packet
2: N = number of available channels
3: channels = [1,...,N]
4: Shuffle channels
5: shortestWaitingTime = ∞
6: for Every channel c in the channels list
7:     t = waiting time on the c channel, taken from DutyCycleHelper
8:     if t < shortestWaitingTime
9:         shortestWaitingTime = t
10:    if t = 0
11:        Send packet to the device's LoraPhy class
12:        Notify DutyCycleHelper of the new transmission
13:        return
14: Cancel previous send event
15: Schedule retransmission of packet after shortestWaitingTime seconds
16: return
```

4.2.3 `LoraPhy`

The `LoraPhy` class models the physical layer of a Semtech LoRa device. Specifically, this is the class that simulates the behavior of the SX1272 and SX1301 LoRa chips in EDs and GWs, respectively. When the device has to send a message, this class's role is to take the packet from the MAC layer and deliver it to the channel 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.

The class makes use of an `m_state` attribute, representing the state of the chip, that can take one of the following values:

- `TX` when transmitting a packet;
- `RC` during reception of an incoming packet;
- `IDLE` when listening for incoming packets;

- `SLEEP` when in low power consumption mode.

Each one of the states above can be linked with a different voltage and current consumption by the `energy` model, which takes note of the energy expenditure of a device and can subsequently give an estimate of the device's battery life. Even though the energy model has not yet been fully integrated in the simulator, particular care was taken in the design of the classes so that combining the `lora` and `energy` models in future iterations of this work will be trivial.

Just like it was done with the design of the `LoraMac` classes, `LoraPhy` also features two subclasses, `EndDeviceLoraPhy` and `GatewayLoraPhy`, which represent in greater detail the PHY layers of EDs and GWs. Both implementations compute interference similarly, via a `LoraInterferenceHelper` class that keeps track of every signal that arrives at the device, and then uses this knowledge to compute whether or not a given packet was impaired by interference.

Algorithm 4.2 shows the steps that are taken by an `EndDeviceLoraPhy` object whenever it is notified by the `LoraChannel` class that a signal is arriving at the antenna. First of all, the object performs an `Add` call to its `LoraInterference-Helper` instance, so that the helper will create an instance of the `Event` class to represent the signal, and add it to a list that keeps track of all events (i.e., packets) that arrived at that device. An `Event` holds all the information that is needed to perform interference computations, namely the time window during which the packet was impinging on the antenna, its spreading factor, received power and logical channel it was sent in. After the packet is kept track of by the interference helper, its power is compared with the device's sensitivity to that spreading factor, and an end of reception event is scheduled only if the device is, in fact, able to receive the packet. Furthermore, the procedure above is performed only if the underlying PHY is in its IDLE state: in case the device is either sleeping, receiving or transmitting, reception of another packet is not possible. It must be noted that, even if the signal is under the sensitivity, an event is still added to the list of interferers, since it will still need to be accounted for when computing Eq. (3.22). This reasoning also applies to the state of the device: even if a device is sleeping, arriving signals must still be registered since they would become influential if the device were to wake up and start listening before the signal ends.

Procedure 4.2 Beginning of reception of a packet.

Input: packet = payload of the packet that is being received
Input: sf = spreading factor of the signal
Input: sensitivity = minimum power, in dBm, that is needed to receive a packet of SF = sf
Input: rxdBm = reception power of the signal
Input: d = duration of the signal

```
1: Notify the LoraInterferenceHelper of the impinging signal with spreading factor sf
2: if state = IDLE
3:     if rxdBm < sensitivity
4:         return
5:     else
6:         Switch to the RX state
7:         Schedule the end of the reception of the packet after d seconds
8: return
```

The end of reception procedure is shown in Algorithm 4.3: first of all, the `EndDeviceLoraPhy`'s `LoraInterferenceHelper` instance is queried to ascertain whether the current packet was destroyed by interference or not. Based on this function call's result, the device either forwards the correctly received packet up the stack or does not, before going into SLEEP mode in either case. The procedure that is used to determine whether a packet suffers fatal interference by other signals can be seen in Algorithm 4.4: the set of interfering signals is grouped according to the spreading factor, then for each SF the cumulative interference energy is computed as the sum of the energies from each signal. Energies are obtained as the product of received power and overlapping time with the intended signal.

The implementation of the `GatewayLoraPhy` class is slightly more complicated than that of `EndDeviceLoraPhy`, since multiple receive paths need to be implemented as depicted in Figure 2.3. In order to do this, the `GatewayLoraPhy` class features as a member variable a list of `ReceptionPath` objects. This class represents a receive path: its variables indicate whether or not it is receiving an `Event` and the channel it is listening to. When the `StartReceive` method of a

Procedure 4.3 Ending of reception of a packet.

Input: packet = payload of the packet that is being received

Input: sf = spreading factor of the signal

Input: rxdBm = reception power of the signal

Input: d = duration of the signal

1: lost = whether packet was destroyed by interference or not (as determined by `LoraInterferenceHelper`)
2: **if** lost
3: Switch to SLEEP state
4: **else**
5: Pass the correctly received packet up to the `LoraMac` class
6: Switch to SLEEP state
7: **return**

`GatewayLoraPhy` is called by the `LoraChannel`, the procedure is similar to that represented in Algorithm 4.2, but can be performed only if the receive path list contains an instance of `ReceptionPath` that is free and listening to the channel of the incoming packet. If the arriving transmission is above the sensitivity, the receive path is marked as busy and linked to an `Event` representing the signal. Similarly, the `EndReceive` procedure is similar to 4.3, but contains some added steps to free the receive path that was linked to the incoming transmission.

The `LoraPhy` classes are also the location where packet tags are applied. In NS3, a packet tag is a customizable data structure that holds some kind of information regarding a packet, and that can be attached to one. More specifically, in the case of this thesis a `LoraTag` class was created. This tag is built to contain information about the spreading factor that is used by a packet and, in the case in which a packet is destroyed by interference, about the spreading factor that caused the loss. This way, whenever a packet is lost at the PHY layer due to interference the packet tag is changed, so that the simulation script knows what happened to every single packet in the simulation.

4.2.4 LoraChannel

The LoraChannel class models the wireless channel that is shared by all devices in a LoRaWAN: this class is responsible of taking packets that a PHY layer wants to transmit and delivering them to a set of LoraPhy objects, with a receive power that is computed according to the Link Measurement Model described in Section 3.2. During the configuration phase, LoraPhy objects are "added" to the channel object, which registers them in a list. The LoraChannel class allows interconnection between registered PHYs via two methods: Send and Receive. When a PHY needs to send a message in the channel, it can do so by calling the Send function with parameters such as the spreading factor of the message, the PHY-layer packet itself, the duration, transmission power and channel number. Upon calling of this method, the channel goes through the list of PHYs, and schedules a Receive event for those nodes that are registered as listening to that communication's channel. In order to determine the time at which to schedule a given Receive event, the channel uses a PropagationDelayModel, a default NS3 abstract class that, given the MobilityModel (i.e., the positions) of the transmitter and the receiver can compute the delay according to various models. In this thesis, the delay model that was used is the ConstantSpeed-PropagationDelayModel, which simply computes the time of flight given the distance between the two devices. The Receive event is scheduled with a set of parameters that the target PHY needs to know in order to perform the set of function calls described in Procedure 4.2: signal duration, spreading factor, channel and power at the receiver location. This last parameter is computed by the LoraChannel by resorting to the PropagationLossModel, an object which computes the power loss based on the transmit power and the locations of transmitter and receiver. The PropagationLossModel used in the simulations is composed by the concatenation of three loss models, described in the following paragraphs.

The first model is represented by the LogDistancePropagationLossModel class, which computes the external path loss according to Eq. (3.8). This model is available by default in NS3, and only needs to be configured to use the appropriate parameters.

The CorrelatedShadowingPropagationLossModel consists in a class implementing the correlated shadowing that was built specifically for the purposes of

this thesis. The class works by dividing the simulated two dimensional space in a grid. Whenever a request is made to the class to compute the shadowing for a transmission going from a point a to point b, it checks whether the square of the grid containing the a position already has a shadowing map associated to it. If a shadowing map is not found, a new `ShadowingMap` object, consisting in an empty grid of values, is created. This object, then, is queried to find the shadowing at position b. The `ShadowingMap` instance computes the positions of the four vertices of the grid around b and, if no values were created for those points yet, it generates shadowing values for those positions according to a normal random variable with the appropriate variance. After this step, the shadowing value for the b point is interpolated according to the procedure described in Section 3.2.3.

Finally, the `BuildingPenetrationLoss` model is a custom class that implements building losses according to [3], by leveraging data structures about buildings, obtained using the default class `BuildingsHelper`. `BuildingsHelper` is an object that holds a map of all the buildings that were created in a simulation, and can then attach to a node information on whether it's inside or outside a building, and at which floor, based on the list of buildings and the node's position. This information is then leveraged by the `BuildingPenetrationLoss` class to determine the losses the transmission experiences because of buildings. The loss for transmissions between a pair (a, b) of devices is computed as the sum of the three components described in Eq. (3.12), counting the contributions of both devices: if a and b are both indoors, the total loss will be considered equal to the sum of two external wall losses, two inner wall penetration losses and two gains due to floor height. The specific values are computed according to 3.2.2, and remain the same for the same device throughout the simulation.

One great advantage of using a simulator such as NS3 is that each one of these three loss models can be switched off or substituted with another one in an extremely simple way. This also makes the evaluation and comparison of many different propagation models easier, since many of the classic models are already implemented in NS3 by default. Once the total loss for the given link is computed, the `Receive` event is scheduled, and simply consists in calling the `StartReceive` method of the `LoraPhy` instance with the appropriate parameters.

During a simulation, only one instance of the `LoraChannel` class is created, and all PHY devices are connected to it. It's important to notice that `LoraChannel`

is completely unaware of issues such as sensitivity and interference: its only task is computing the received power and delay at the location of every other device in the network, given the transmit power and position of the transmitter. In order to speed up the simulations featuring nodes that span a very large area, a power threshold can be introduced so that every transmission that would arrive under that threshold value won't even be delivered to the PHY layer. Additionally to this measure, a way to greatly reduce the computational time needed for the simulation of UL only networks is to check whether the PHY device the channel is delivering the message to is an ED or a GW, and only performing the necessary computations of the received power and schedule the receive event if the receiving PHY is a GW. In fact, it's useless to perform interference computation at the end devices if all traffic is expected to flow in the UL direction, since an ED will never open a receive window.

4.2.5 `LoraNetDevice`

The `LoraNetDevice` class models a "LoRa network card": this `NetDevice` can be attached to a `Node`, whose applications can then use the card to send data to other LoRa devices. A `LoraNetDevice` is essentially used to hold together all the LoRa objects that need to be aggregated to a node: a `LoraPhy` and a `LoraMac`. One consideration must be made regarding the abstract class `NetDevice`, and its orientation towards IP communications. It is stated in the `LrWpanNetDevice` API documentation that "The ns3::NetDevice includes IP-specific API such as GetMulticast, Send and SendTo methods, which do not map well the the 802.15.4 MAC MCPS DataRequest primitive. So, the basic design is to provide, as much as makes sense, the class ns3::NetDevice API, but rely on the user accessing the LrWpanMac pointer to make 802.15.4-specific API calls". Likewise, the `lora` module uses the NetDevice as an encapsulating class, and only leverages a generic Send version that is adapted to handle the underlying MAC layer. No support for concepts such as multicast and IPv6 addresses is provided.

4.2.6 Other Classes

A set of additional classes were written in order to manage the simulations more easily.

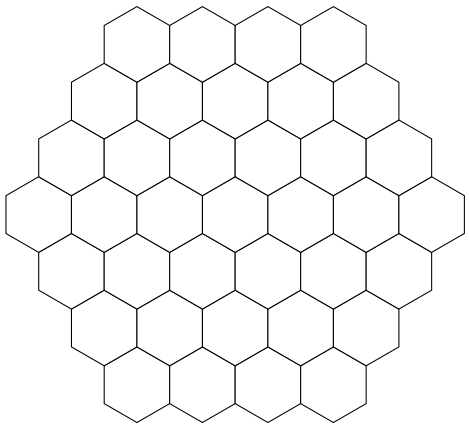


Figure 4.2: An hexagonal grid as generated by `HexGridPositionAllocator`.

The `HexGridPositionAllocator` is a class that can be used to compute the positions of gateways so that they are placed in an hexagonal grid layout. The class works by allocating the first gateway at the center of the simulation space, and then creating "rings" around the central gateway by following the desired pattern of an hexagonal grid. The obtained disposition is shown in Figure 4.2, where a gateway is placed at the center of each hexagon. Given a hexagonal grid with a number n_r of rings, we can compute the total number N of gateways it contains with the following formula:

$$N = 3n_r^2 - 3n_r + 1. \tag{4.1}$$

An helper class called `LogicalLoraChannelHelper` was created to move the channel management logic outside of the `LoraMac` implementations. This class holds a list of `LogicalLoraChannel` objects, each one characterized by a frequency, bandwidth and channel number. This class is supposed to be used to manage the channels that are available for communication, and to add and remove them as a response to the appropriate MAC commands in a future iteration.

`LoraMacHeader` and `LoraFrameHeader` are subclasses of the `Header` class, which is used to represent the header of a `Packet`. These classes support serialization and deserialization of a series of fields that depend on the protocol at hand. While in the current iteration of this thesis's work the headers are zero

filled to reduce the system complexity, in future iterations they will be used to transport MAC commands, the standard major version number, addresses and other pieces of information as specified in 2.3.2.

4.3 Helpers and Tests

Aside from the fundamental building blocks described above, a set of "helper" classes was also implemented to make configuring a Lora network easier. Helpers are an NS3 design element, intended to assist script creators in setting up topologies and nodes that are fully configured to use the desired module.

Whenever an instance of the `PeriodicSender` application is created, its period must be set. In order to make correct configuration of many of these classes easier, applications can be deployed on a set of nodes by using the `PeriodicSenderHelper` class. This helper decides a node's reporting period according to the [3] specification, so that the appropriate distribution of periods among nodes described in Table 3.4 is respected.

Another set of helpers were then written in order to appropriately install and configure the Lora stack at once on a given set of nodes: the classes `LoraHelper`, `LoraPhyHelper` and `LoraMacHelper` were designed to work in synergy to create and deploy `LoraNetDevice`, `LoraMac` and `LoraPhy` objects on a wide set of nodes, making sure that layers are configured to communicate appropriately with one another and that the PHY layers are correctly connected to the `LoraChannel` instance they share.

A set of tests were also written, along with the module, in order to ensure the correctness of the software after each update and to comply with the NS3 guidelines. Elementary message delivery is tested to ensure that the PHY layer at an end device is able to receive a message from a device within range. Channel separation tests make sure that a device listening to channel i will not receive communications sent in channel j when $i \neq j$. Interference checks verify that a packet can be destroyed by an interferer with sufficiently high power. Furthermore, they also verify that communications on different channels do not interfere. Finally, gateway parallel decoding capacity tests make sure that a gateway can receive up to 8 and no more than 8 messages in parallel.

Procedure 4.4 Determine whether a packet was destroyed by interference or not.

Input: packet = the packet we are interested in receiving
Input: duration = the duration of the packet we are interested in receiving
Input: interferers = list of interferers registered so far
Input: rxPowerDbm = power of the packet
Input: sf = spreading factor of the packet

```
1:  packetLost = false
2:  for currentSf ∈ 7, . . . , 12, until packetLost = false
3:      cumulativeInterferenceEnergy = 0
4:      for interferer ∈ interferers
5:          if interferer's channel number = packet's channel number OR interferer = packet
6:              Skip this event
7:          Remove the current interferer from the list if it is older than a threshold
8:          if interferer's sf = currentSf
9:              overlap = overlap between the interferer and packet windows
10:             interferenceEnergy = overlap * interferer's power
11:             cumulativeInterferenceEnergy += interferenceEnergy
12:     if cumulativeInterferenceEnergy ≠ 0
13:         snir = snir computed according to Eq. (3.22)
14:         snirIsolation = isolation from Eq. (3.20), according to sf and currentSf
15:         if snir ≥ snirIsolation
16:             Continue with the loop
17:         else
18:             packetLost = true
19: return packetLost
```

5 Performance Evaluation

This chapter covers some simulations that were performed using the module described in Chapter 4, and analyzes the results. After a brief introduction covering how the network was deployed and some of the techniques that were used to set up and speed up the simulations, metrics such as gateway coverage, throughput and successful packet delivery rates are evaluated in a variety of scenarios.

5.1 Simulation Scenario

In order to simulate a LoRaWAN, the NS3 simulator needs to be properly configured. The single components of the `lora` module described in Chapter 4 model various aspects of a Lora network, however the individual classes must be properly combined one with another in order to simulate a network. To do so, a simulation script that leverages the system of helpers to create and configure a great number of devices is necessary. Furthermore, the simulation script is also used to gather data from the simulation, leveraging the trace sources that were placed inside some significant class variables: whenever a certain event happens during the simulation, a call to a script-level function is performed, so that the event can be registered in one of the script data structures to be analyzed later on.

5.1.1 The Simulation Script

The main steps taken by the simulation script are described in Procedure 5.1. After the creation of the `LoraChannel` class and of the related propagation loss models, the nodes representing EDs are created and assigned a uniformly random position inside a circle of radius r, using the default NS3 class `UniformDiscPositionAllocator`. Then, the `LoraHelper` class is used to configure the LoRaWAN stack on the end devices, and to connect them to the channel instance. For all the simulations performed in this thesis, message ACKs were disabled: this has the twofold effect of simplifying the simulation and allowing some features that speed up simulations. EDs are preconfigured to be able to communicate in the network, so that there is no need to perform any join procedure. Gateway nodes are then created, allocated in a hex grid and configured to use the LoRaWAN stack and the receive paths allocation specified in 3.3. Callback functions are also connected to the trace sources in the code to collect information about changes in the state of the simulation. Both EDs and GWs are configured to have a fixed position during the whole simulation. At this point, a set of buildings is created so that it spans the entire simulation area. The buildings are laid out as a grid of rectangles, inspired by the layout of the Manhattan area, with each building having a size of 130 by 64 m, and distances between two buildings of 32 and 17 m for streets that go from North to South and from East to West, respectively. An example of the building layout can be seen in Figure 5.1, where buildings are represented as grey boxes. After the buildings have been created, the propagation loss model is ready to be used. This information is used to set up each ED's spreading factor, according to the procedure described in detail in Section 5.1.3. After this step, an optional step that consists in pruning devices that cannot reach the gateway because of heavy shadowing and distance can be performed. As a last configuration step, the `PeriodicSender` applications are installed on every ED and set up to start and stop at fixed times. Finally, the simulation is started and, after it ends, the script collects the results from its data structures and saves them on a file or displays them as output of the program.

Procedure 5.1 Simulation setup script.

Input: r = simulation radius

1: Create the `LoraChannel` object, configure it to use the delay and loss models
2: Create the nodes representing the EDs
3: Assign each ED a uniformly chosen random position inside a circle of radius r
4: Install a Lora stack on each ED node, connect its PHY to the channel
5: Connect the callbacks of the ED trace sources to local functions
6: Create the nodes representing the GWs
7: Assign each GW a position according to `HexGridPositionAllocator`
8: Install a Lora stack on each GW
9: Configure the receive paths on the GWs
10: Connect the callbacks of the GW nodes
11: Create buildings
12: Set up the EDs' SFs
13: Prune EDs
14: Set up applications on EDs
15: Start the simulation
16: Save the simulation results
17: **return**

5.1.2 Variables and Metrics

The simulation described above requires the definition of many different variables. These can be adjusted to see their effect on the performance of the system as a whole. Some notable variables are:

- Network scale: higher numbers of EDs in a network will yield higher probabilities of two signals interfering. The number of gateways that are deployed on a fixed area, on the other hand, will affect the coverage of devices inside buildings: a higher gateway density will allow devices that experience heavy shadowing to communicate.

- Model of the traffic generated by the application layer. At a constant device density, lower message interarrival times will cause more collisions between packets. Synchronicity in packet departure times could as well increase the

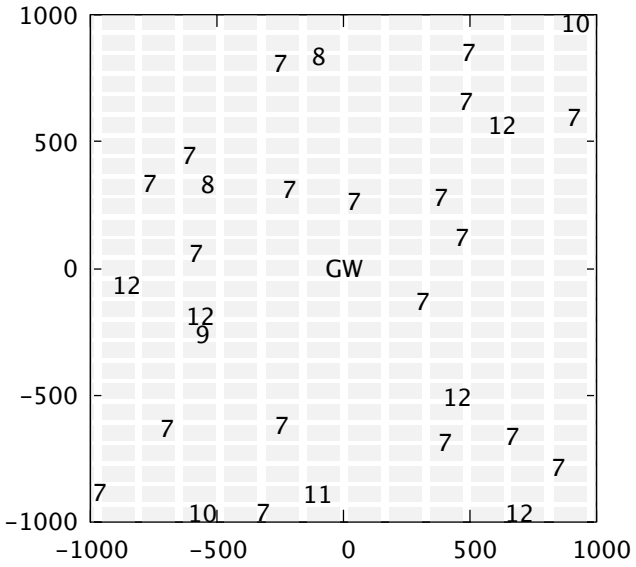


Figure 5.1: An example of random distribution of nodes around a gateway.

probability of destructive interference events.

- Components of the path loss model can affect the gateway coverage range: a smaller shadowing variance or an assumption that considers "lighter" external walls in buildings decreases the loss computed by the channel, and enables a transmitter's signal to travel farther before falling under the sensitivity of the receiver.

Likewise, many different metrics can be evaluated in a LoRaWAN system:

- Spreading Factor distributions: how spreading factors are distributed at different distances from the gateway, and how this is affected by the propagation loss model in use.

- Lost messages: application layer messages can be dropped because of duty cycle regulations, and PHY layer messages can be lost because they arrive at the gateway with a power under sensitivity or because of destructive interference. The rate at which packets are lost can be investigated as a function of a variable, in order to highlight a specific mechanic of the system.

- Throughput of the network: considering the system as a whole, it may be interesting to look at the raw data extraction rates that can be achieved. In this case, the focus will be on network efficiency, and not on a typical device's probability of successfully delivering a packet.

5.1.3 Spreading Factor Assignment

Before starting a simulation, each device is assigned a SF according to the procedure described in Algorithm 5.2. We first calculate the power level that each gateway would receive from the end device. Then, we pick the gateway with the highest received power and set the SF based on that value. The assignment is done according to the gateway sensitivity: we assign the end device the lowest SF that would still be above the gateway sensitivity. This is done in order to minimize the ToA for that device's packets, and thus lower the probability of collisions. Note that, due to the shadowing and the presence of buildings, the closest gateway to a device may not always be the one that receives the highest power from that device. This procedure is performed so that no further ADR adaptations are needed. Furthermore, while in a real network shadowing conditions would change with time, in the case of this thesis the channel was considered to be time independent, so once SFs are set up there is no need for adaptation.

As an example of this procedure, suppose that the best gateway for a device receives a power of -137 dBm. In this case, considering the sensitivity values contained in Table 3.2, it can be seen that $SF = 9$ would be too low, while we can receive the end device's packets if they are sent using $SF \in \{10, 11, 12\}$. Since we are interested, in general, in minimizing the ToA, we set the end device to use $SF = 10$.

The SF assignment described above can be visualized in Figure 5.2, showing a portion of a simulated deployment for different propagation models. One single gateway was placed at coordinates $(0, 0)$. EDs are represented as crosses and a device's spreading factor is encoded as a color. In Figure 5.2a it can be seen that, in case only the Log-Distance propagation loss model is used, the distance of a device from the gateway is the only component that determines the SF: devices up to 3500 meters away from the gateway are configured to use SF 7, while farther distances force devices to use higher spreading factors. The maximum radius

Procedure 5.2 Spreading factor assignment.

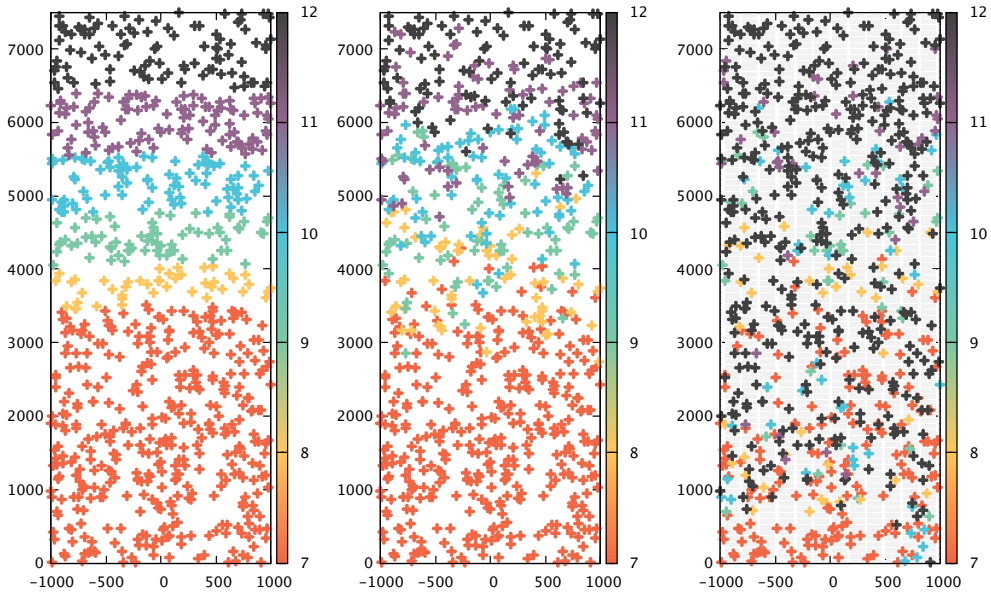
Input: S_i = GW sensitivity to SF i

1: **for** Each ED e in the network
2: **for** Each GW g in the network
3: Compute the received power for the transmission from e to g
4: Select the GW that received the strongest signal, having power P
5: Assign to e the lowest SF such that $P > S_i$
6: **return**

a device transmitting at 14 dBm ERP can achieve with the model described in Eq. (3.8) is approximately 7.5 km. When shadowing is added to the propagation model, as shown in Figure 5.2b, the borders between the sets of devices that use the same SF become blurrier: some devices that would be slightly above sensitivity get a worse link because of shadowing, while others don't get penalized as much. Finally, Figure 5.2c shows the effect of adding buildings to the simulation: some devices experience heavy shadowing even if they are within 1 km of the gateway, and are forced to use higher spreading factors to contrast the building penetration losses. This situation can also be visualized in Figure 5.1 in greater detail. On the other hand, those devices that aren't placed inside a building experience the same losses as they experienced in Figure 5.2b.

5.2 Results

This section analyzes various metrics obtained thanks to the `lora` module that was added to NS3. First of all, some results on the network's throughput performance are evaluated. Then, a network is set up with a realistically modeled environment and traffic model in order to evaluate the probability of losing a packet in a LoRaWAN. Some experiments were then performed to evaluate the performance of each single spreading factor on interference, and how packet success rate changes with spreading factor. Finally, some tests on coverage in a realistic setting were performed, in order to evaluate the gateway density that is required to cover a city with deep shadowing caused by buildings.



(a) Propagation only. **(b)** With shadowing. **(c)** With shadowing and buildings.

Figure 5.2: Spreading factor distributions for different propagation models.

5.2.1 Throughput Performance

The first simulation campaign aims at evaluating the network throughput S as a function of the network offered traffic G. The scenario is characterized by a single central GW and N EDs that are uniformly distributed in a circle of radius $r = 7500$ m around it. This particular radius value was chosen because r is the maximum distance at which the gateway and an end device using $SF = 12$ are able to communicate above sensitivity only considering the propagation loss. The simulations for this section have been performed on a single logical LoRa channel. The gateway was configured to only have one receive path enabled for all simulations measuring throughput.

For the computation of throughput, we suppose that EDs $i = 1, \ldots, N$ generate every τ_i seconds a packet which occupies the channel for $t_{p,i}$ seconds in order to be transmitted. Note that for the simulations described in this section, unless stated otherwise, duty cycle limitations were not applied. In fact, we are interested in testing the LoRaWAN access scheme in itself, regardless of the local regulations that may modify its performance. We compute the network offered traffic as

described in [39]:

$$G = \sum_{i=1}^{N} \frac{t_{p,i}}{\tau_i} \,. \tag{5.1}$$

The offered traffic is, in other words, a measure that expresses the fraction of time the channel is occupied by transmissions by end devices. $G < 1$ means that the channel is underutilized, since there are times at which no transmission is going through the channel. On the other hand, $G > 1$ means that, even with a perfect synchronization of the devices, some packets will necessarily try to use the channel at the same time, causing a collision.

For a given value of G, throughput S is then obtained as:

$$S = G \cdot P_{\text{succ}} \,, \tag{5.2}$$

where the probability of success of a given packet P_{succ} is approximated as the ratio between the number of successfully received packets and the total number of sent packets during a simulation. A network offering a traffic of 1 and featuring perfect synchronization between devices, so that collisions are avoided and no packets are lost, will yield a throughput of 1. Of course, a perfect synchronization between devices is impossible to achieve, so it is expected that $S < 1$. Especially for networks where devices access independently the channel with no coordination such as time or frequency slotting, as is the case of LoRaWAN, throughput is expected to follow the shape of the ALOHA medium access protocol.

As a first validation of the simulator described in Chapter 4 we expect that, under ideal channel conditions, the shape of the throughput curve for a varying offered traffic will be that of a typical ALOHA network. If we turn off the link measurement model, all end devices are configured to transmit with $SF = 7$ and all packets have the same ToA (provided the payload length is fixed) and are received with the same power at the central gateway. We also assume that overlapped packets always fatally collide and are, therefore, lost: this translates into using a T matrix that is filled with very large numbers, so that at the slightest overlap between two packets there is destructive interference for both transmissions. In fact, if we used the interference matrix of Eq. (3.20), packets with the same SF would have an isolation of 6 dBs, and slightly overlapping packets may survive the collision as was the case for extremes offsets shown in

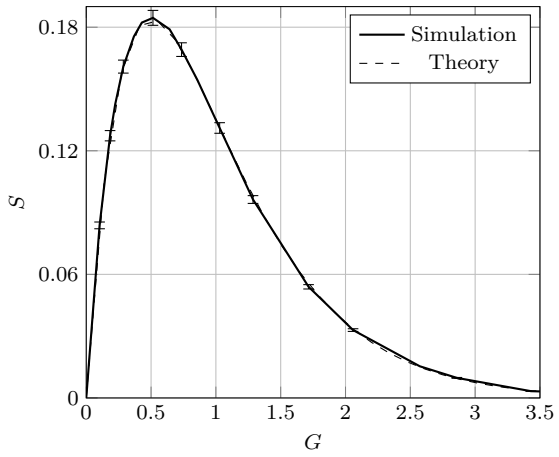


Figure 5.3: Throughput for $SF = 7$ and ideal packet collisions.

Figure 3.1. Using this setting, the expression for the traffic offered by a network of N devices can be expressed as:

$$G = \frac{N \cdot t_7}{\tau_i} \,. \tag{5.3}$$

where t_7 is the ToA for a packet using SF 7 and a fixed payload length.

As expected, the performance result of this test, shown in Figure 5.3, follows the typical ALOHA throughput trend [39], for which an exact expression exists:

$$S = G \cdot e^{-2G} \tag{5.4}$$

After this first validation, we can evaluate the impact of using a variety of SFs and a real wireless channel, by using the log-distance component of the proposed link measurement model: indeed, the presence of a real channel motivates the usage of all possible SFs to allow farther devices to communicate with the GW. The fact that multiple SFs are used also allows the network to leverage the quasi-orthogonality of transmissions using different SFs, using the collision matrix T in its Eq. (3.20) version. The simulation results of Figure 5.4 show a large throughput increase with respect to the previous case.

We also studied the impact of $SF = 12$ transmissions on the performance of a LoRa network, especially on interference. In order to do so, a simulation

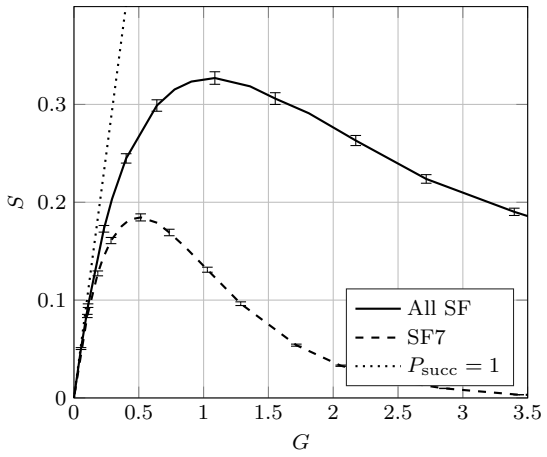


Figure 5.4: Throughput performance of a LoRa network with real wireless channel (solid line) and without it (dashed line).

was created in which EDs configured to use SF 12 were not allowed to transmit, even though their traffic still counted towards the generated traffic total. This way, while some of the generated packets will always fail transmission and count as lost packets in the P_{succ} ratio, they will also generate no interference with other transmissions in the network. The simulation results shown in Figure 5.5 demonstrate that excluding end devices with the highest SF is beneficial when the system load is high, because the collisions with other end device transmissions are reduced, and thus the success probability increases up to $S_{\text{gain}} = 0.12$ with respect to the scenario in which SF 12 devices are allowed to transmit. On the other hand, for low offered traffic values, when interference is not the limiting factor for throughput, the fact that some packets are not transmitted affects throughput in a negative way, causing a loss of up to $S_{\text{loss}} = 0.1$. This behavior is in line with the mandate by the LoRa Alliance to exclude from public networks end nodes which only transmit at $SF = 12$ and refuse to change their SF: while these nodes will have higher probabilities of successfully delivering a packet given the better sensitivity of the GW to SF 12, their effect on the network as a whole would be detrimental.

Finally, we investigated the effect of applying duty cycle restrictions to the end device's transmissions. Figure 5.6 shows the effect of limiting an ED's packet transmissions to a fixed, maximal rate that respects a 1% duty cycle. This rate

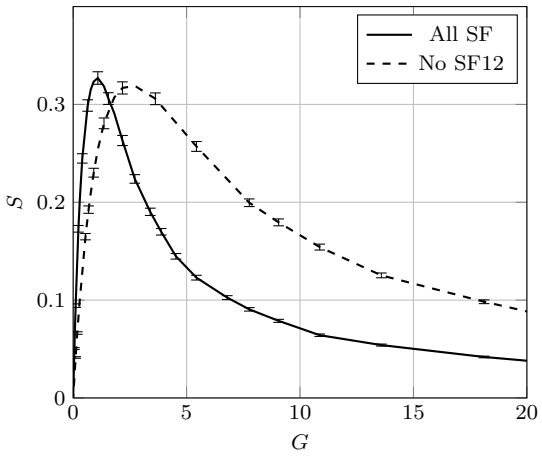


Figure 5.5: Throughput comparison with and without $SF = 12$.

depends on the SF employed by a node, since higher SFs will use the channel for longer when compared to lower SFs: since G is computed based on the same transmission period for all devices, at a fixed G value some of the devices in the network may be limited by the duty cycle, while others are still transmitting under that limit. Figure 5.6 shows that a duty cycle of 1% is actually beneficial for the system, because it limits traffic and hence collisions. This allows the throughput, after experiencing a drop in performance up until all nodes in the network are under the effect of duty cycle limitations, to stabilize at a fixed value of $S_{1\%} = 0.14$. This stabilization counters the continuous drop in performance that would follow an increase of the offered traffic if no limitations were applied, and can ensure a gain in performance for a very small additional complexity in the ED software.

Confidence intervals were computed for all plots shown above, and confirm the gains in performance that can be achieved both by removing SF 12 devices and by enforcing a duty cycle.

5.2.2 Success Probability Performance

The second simulation campaign aimed at estimating the probability of successfully receiving a packet in a LoRa network. Since we are interested in the performance of real networks, this simulation scenario features 36 gateways that are

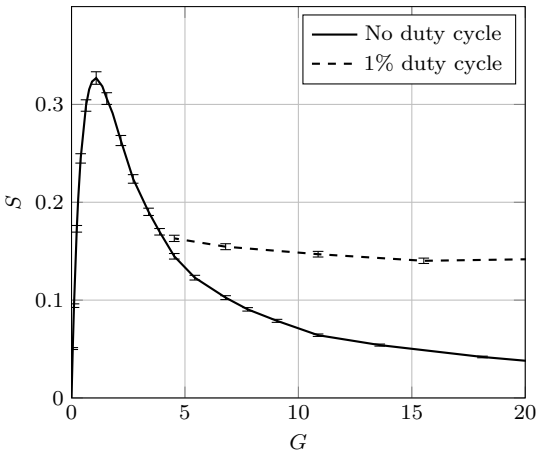


Figure 5.6: Effect of duty cycle limitations on throughput.

placed in an hexagonal grid around a central gateway, emulating a real system with cellular-like coverage of a large are. In these simulations each gateway will cover a radius of 1.5 km, thus the area in which we place end devices is a circle of radius 7.5 km, centered on the central gateway. Such a big network allows us to simulate inter-cell interference besides intra-cell interference: even though the simulation features 37 gateways, we are only interested in the devices belonging to the area that is covered by the central gateway, so the collected data regard packets that were generated inside this region of interest. To add realism to the simulation, the entire area contains buildings and generation of correlated shadowing is enabled. If a device's position is randomly assigned to an area occupied by a building, that device will be marked as "indoor" and transmissions involving it will suffer from building penetration losses. As for the traffic generation, we refer to the Mobile Autonomous Reporting periodic reports model described in [3]. Also the size of the application level payload is randomized, following a Pareto distribution as described in [3] with payload size in the $[10, 30]$ bytes range. This set of parameters allows us to model a realistic network, however the simulation of such a big network calls from some optimizations in order to speed up computations.

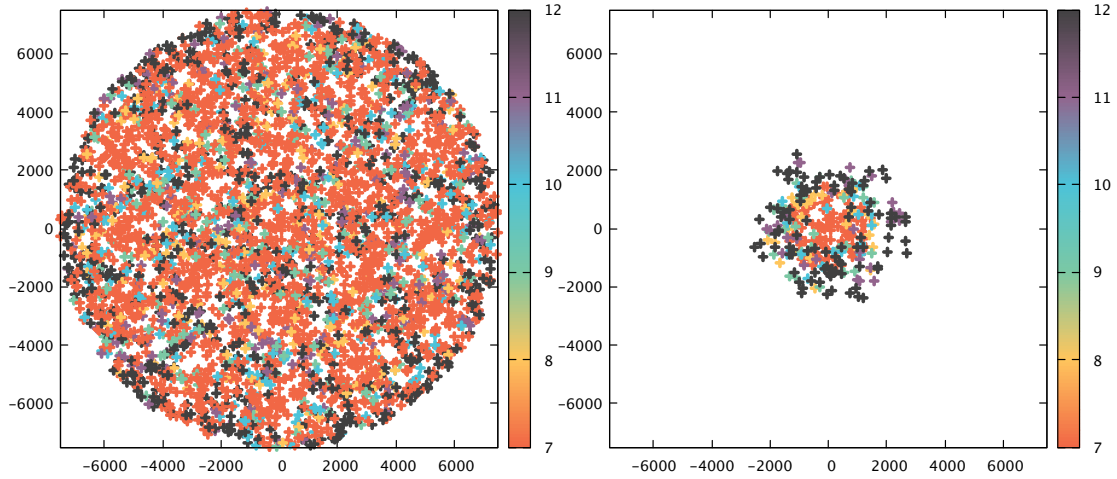
Procedure 5.3 Pruning algorithm.

Input: radius = radius of the simulation

```
1:  r = 0
2:  exit = false
3:  for sf ∈ 7, . . . , 12
4:      for r < radius, exit = false
5:          insideEnergy = outsideEnergy = 0
6:          for Each end device e
7:              if e's SF = sf
8:                  energy = e's energy for a transmission
9:                  if e's distance from center < r
10:                     insideEnergy = insideEnergy + energy
11:                 else
12:                     outsideEnergy = outsideEnergy + energy
13:         if outsideEnergy < insideEnergy/10
14:             exit = true
15:         else
16:             r = r + ϵ
17: return
```

Pruning of End Devices

An approximation that helps speeding up the simulations involves pruning those end devices that would yield an irrelevant interference at the central gateway, according to the following criterion. Let's call $\mathcal{I}_r$ the set of devices that are inside the circle of radius r and centered on the central gateway, and $\mathcal{O}_r$ the set of devices outside this circle. Let $E_{\mathcal{I}_r}$ be the total energy received at the central gateway because of devices inside the circle, and $E_{\mathcal{O}_r}$ the energy delivered by devices outside the circle. After finding the smallest r for which $E_{\mathcal{I}_r} \ll E_{\mathcal{O}_r}$, we prune all devices belonging to $\mathcal{O}_r$, detaching the PHYs from the channel and removing the nodes from the simulation. The procedure is described in Algorithm 5.3, and the effects can be seen in Figure 5.7: the fact that simulated devices span an area that is larger than the central gateway cell allows us to model the effects of



(a) All devices.

(b) Devices left after pruning.

Figure 5.7: Effects of the pruning algorithm.

inter-cell interference, besides intra-cell interference, in an efficient way.

Figure 5.8 shows the packet success probability as a function of the number of end devices in the central gateway coverage area. This probability ignores packets that arrive at the central gateway under sensitivity because of heavy building loss or shadowing, thus the declining probability of success is to ascribe only to interference or to the unavailability of adequate reception paths at the gateway. The percentage of nodes that were unable to reach a gateway with a sufficient power was, in the case of this scenario, 20%. The simulation spans 1 day and was repeated for 10 times, to simulate different shadowing and ED placement scenarios. Computation of confidence intervals assure us that this was, indeed, a large enough sample. The decreasing trend appears to be linear with the number of devices in the network, with a success probability around 96% for a gateway tasked with serving 14000 EDs. This is coherent with Semtech's claim that a gateway is able to support a network of around 10^4 nodes [10], if the target probability of success of a transmission is fixed at 95%.

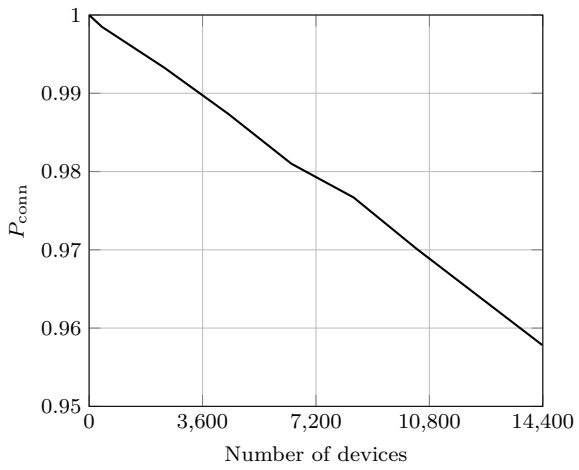


Figure 5.8: Probability that a packet is successfully received as function of the number of end devices.

5.2.3 Spreading Factor Statistics

Another series of simulations was aimed at evaluating the behavior of different SFs in a LoRaWAN. In the case of these simulations, a network of $N = 8000$ devices was deployed in a circle of radius 7500 m. In this case, correlated shadowing and building losses were turned off to ensure that every device was able to reach the gateway with a power above the GW sensitivity to SF 12 packets. This also means that there are only two reasons why a packet may be lost in this scenario: either because of interference or because of unavailability of suitable receive paths. All EDs are served by a single gateway, placed at the center of the circle. Three statistics were evaluated for each SF:

- P_{succ} is the probability that a packet sent with SF i is correctly received.
- P_{int} is the probability that a packet sent with SF i is corrupted by interference.
- P_{nmr} is the probability that a packet sent with SF i was not received because no more receive paths were available.

Figure 5.9 shows the metrics above for a simulation performed using the MAR Periodic Reports application model of [3]. This translates to a network that generates a relatively low amount of traffic, in which interference doesn't have a

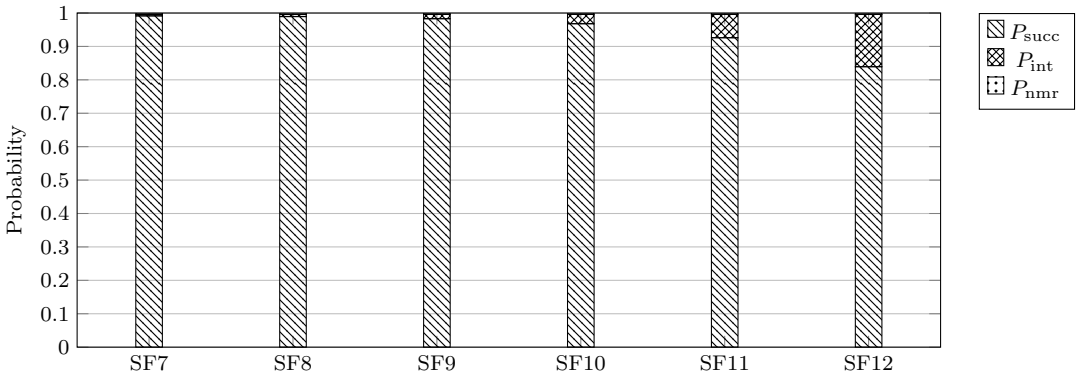


Figure 5.9: SF statistics for a low traffic network.

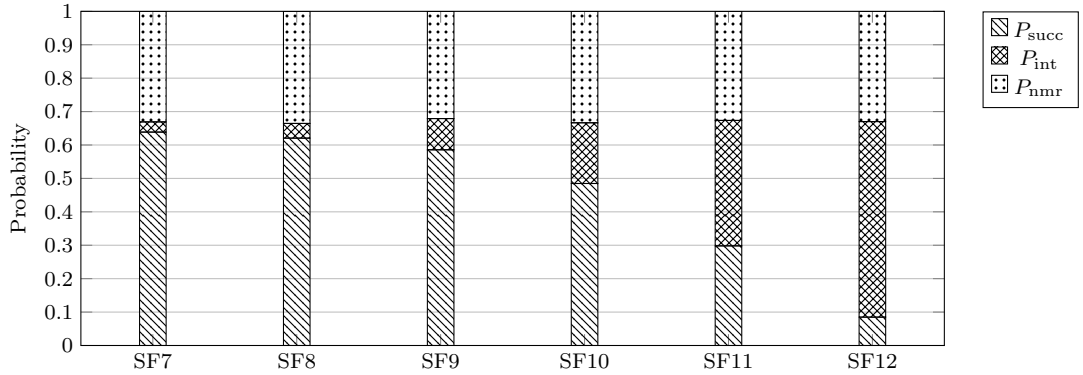


Figure 5.10: SF statistics for a heavily trafficked network.

predominant role: probabilities of success for every SF are above 80%, and above 90% for SF 11 and below. A general trend is observed: the higher a device's SF is, the higher becomes the probability of losing that device's packets because of interference. Since the traffic is low in this scenario, practically no packets were lost because no suitable receive paths were available.

Figure 5.10, on the other hand, represents a situation where the network is experiencing heavy traffic. In this case, EDs were set up to send a packet once every 10 minutes. It can be seen that the trend that was observed in Figure 5.9, where interference increased with SF, is exacerbated in this case by the heavier traffic. The other predominant cause of packet loss is the lack of receive paths in the gateway: this effect is experienced to the same extent by every device, regardless of its SF, since it only depends on the congestion of the system.

5.2.4 Gateway Coverage Assessment

In the final simulation campaign we study how the increase in the number of gateways serving a fixed amount of EDs will enhance the probability that a given node is able to connect to the Network Server. This aspect is particularly interesting for critical applications, where a packet's reception, by any gateway, is crucial. We simulated a circular urban scenario of radius 7.5 km, so that the whole area would still be covered by a single gateway, where end devices are served by an increasing number of gateways deployed in an hexagonal grid setup. The full propagation model was employed to perform these simulations, so a considerable fraction of the EDs experiences heavy shadowing caused by buildings.

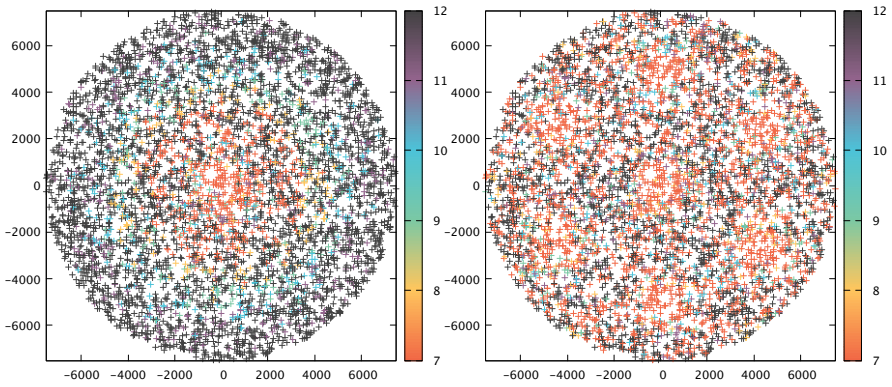
Figure 5.12 depicts the probability that a ED manages to connect to at least one gateway, versus the number of gateways that were deployed in the fixed circle of radius 7.5 km. The simulations show that, in order to achieve a reliability above 90%, gateways should be deployed in such a way that every gateway covers 6 km^2 or, equivalently, has a radius of 1200 m around it. In order to achieve a coverage probability higher than 95%, the distance between two adjacent gateway should be of 2000 m or less. This value can be compared with the figure given in [3] for the radius covered by a CIoT cell, fixed at 577 m: simulations suggest that a LoRaWAN system needs lower GW deployment densities to achieve a satisfactory coverage when compared to base stations in a CIoT scenario.

One of the consequences of increasing the density of gateways is that the number of end nodes with SF $>$ 7 decreases, leaving place to a network that is mostly composed of nodes using SF 7. This phenomenon is depicted in Figure 5.11: a network of devices employing the same SF will see an increased number of collisions between packets, since the diversity of the modulation is not leveraged. In a real LoRa network, the ADR mechanism should be able to keep the network in a state where SF orthogonality can still be leveraged to increase throughput. The performances of such algorithms could, of course, be investigated in the future through the simulator that was developed in this thesis.

5.3 Comments and Further Observations

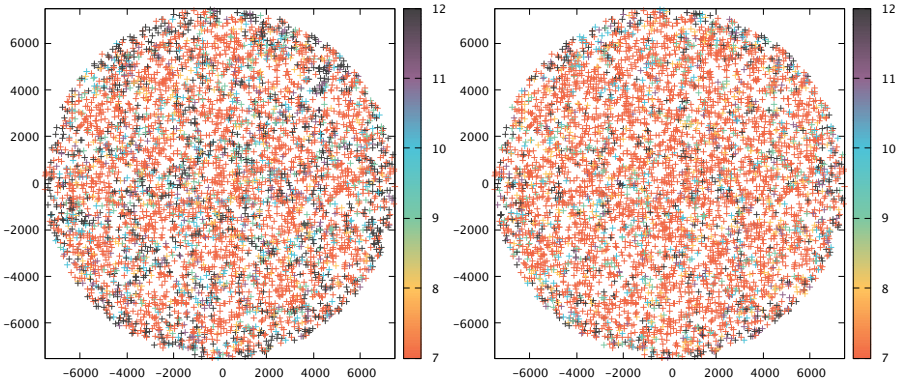
The results derived in this chapter highlight the qualities of the LoRa modulation. First of all it was shown that, by leveraging the quasi-orthogonality between different SFs, a network employing LoRa devices achieves a throughput that is higher than that of a standard ALOHA system, without adding any burden in the MAC scheme, like coordination and synchronization between devices or carrier sensing. This feature allows LoRa networks to scale well, while maintaining a low complexity that is suitable for IoT devices. Furthermore, it's thanks to the modulation and its high sensitivity values that LoRa networks can be deployed in a urban scenario with a reduced density with respect to proposed CIoT networks, as explained in Section 5.2.4. Finally, SFs represent another feature of the network that the NS can tune to optimize the performance of the network, finding the right balance between the resistance to interference brought by SF diversity and the maximum coverage possible.

The fact that the LoRaWAN standard does not define gateway cells means that EDs are not tied to a single gateway. Gateways can be placed freely, and without necessarily following a predefined deployment scheme: this is a key feature that allows crowdsourced solutions like The Things Network to grow so fast, encouraging users to deploy their own gateway and increase the coverage and performances of the whole network. This characteristic, however, leaves room for potential issues, tied to the fact that an ED in an area covered by a certain GW could find easier communicating with the NS via another cell's GW, because of shadowing. This behavior can be seen as an advantage, since it allows devices to find multiple GWs to which deliver their packets, and it avoids the burdens of handover procedures. On the other hand, this mechanic may bring a disadvantage to the GWs that have better exposure. In fact, since the NS forwards DL messages through the "best" gateway (i.e., the gateway that received an UL message with the highest power), a GW placed in a favorable position to cover a wide area may be burdened with DL communications that are intended for EDs that are, in fact, closer to other GWs. This behavior becomes especially relevant in the case of duty cycled transmissions, which apply to GWs too. Solutions to this potential issue could, of course, be evaluated and compared through the framework developed in this thesis.



(a) 1 Gateway.

(b) 7 Gateways.



(c) 19 Gateways.

(d) 37 Gateways.

Figure 5.11: Coverage for different gateway densities.

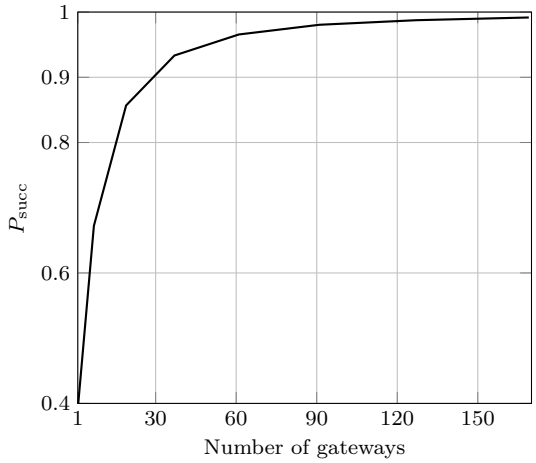


Figure 5.12: Probability of correctly receiving a packet at the Network Server as a function of the number of gateways covering a circular area of radius $7.5\,\mathrm{km}$.

6

Conclusions and Future Work

This work covered one of the most promising LPWAN technologies, LoRa, and introduced a framework that allows the evaluation of network level performance of a LoRaWAN system.

After an introduction that briefly covered the requirements and challenges of the IoT paradigm, the most popular solutions that were proposed to tackle the new demand for massive IoT connectivity were introduced. Then, the LoRa modulation and the LoRaWAN standard were covered, along with the regulatory framework in which these technologies are meant to operate.

A review of the literature on LoRa systems and their performance with regards to range and ability to scale was done. After this introduction to the state of the art, various models of the different components of a LoRaWAN system were described, with particular attention to the modeling of interference and the usage of realistic traffic and propagation models.

After a brief introduction to the NS3 network simulator, the new `lora` extension module, developed as part of this thesis, was described in detail along with the standard NS3 classes that were used to implement some of the models.

After having introduced the characterization of the reference scenario, a campaign of simulations exploring various metrics was performed. Simulation results show that the LoRaWAN access scheme provides a higher throughput with respect to a typical ALOHA-based scheme, thanks to the partial orthogonality between

its spreading factors. Moreover, the LoRaWAN architecture was proved to scale well, mainly due to the fact that an increase in the number of gateways enhances the coverage and reliability of the uplink as well. Finally, a simulation involving a network featuring multiple gateways and a realistic traffic model resulted in a packet success rate above 95% for a gateway serving approximately $15,000$ end devices, confirming some of the claims that were made by the companies supporting this new system.

6.1 Future Developments

As future work the simulator will be extended, in order to allow analysis of further aspects of LoRa networks. An improved simulator would be useful to evaluate the effectiveness of different ADR schemes, simulate different strategies to perform a bootstrap of the network, investigate optimal gateway placement, DL transmissions' impact on the system, frequency planning and co-existence with other networks working in the unlicensed spectrum. Further details, like a CAD mechanism and the capture effect, may also be added to the set of models that has been employed in this thesis in order to better describe how LoRa chips work in reality. Integration with the `energy` NS3 module would allow estimation of battery life for a device in a typical LoRaWAN system, also challenging the claims of an ED being able to survive 10 years on two AA batteries. Another possible area of improvement is the modeling of the NS and of its communication links with gateways, which would allow the identification of bottlenecks and the evaluation of different system management strategies.

References

[1] "Gartner says 6.4 billion connected "things" will be in use in 2016, up 30 percent from 2015," 2015. [Online]. Available: http://www.gartner.com/newsroom/id/3165317

[2] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376.

[3] 3GPP, "Cellular system support for ultra-low complexity and low throughput Internet of Things (CIoT)," Tech. Rep. 45.820 V13.1.0, Nov. 2015.

[4] "The ns-3 network simulator," 2016. [Online]. Available: https://www.nsnam.org/

[5] Ericsson, "White paper: Cellular networks for massive iot," Tech. Rep. Uen 284 23-3278, January 2016. [Online]. Available: www.ericsson.com/res/docs/whitepapers/wp_iot.pdf

[6] M. Centenaro, L. Vangelista, A. Zanella, and M. Zorzi, "Long-Range Communications in Unlicensed Bands: the Rising Stars in the IoT and Smart City Scenarios," *IEEE Wireless Communications*, vol. 23, Oct. 2016.

[7] Semtech Corporation, "Semtech Corporation Website." [Online]. Available: http://www.semtech.com

[8] LoRa Alliance, "LoRa Alliance Website." [Online]. Available: https://www.lora-alliance.org/

[9] Semtech Corporation, "AN1200.22 LoRa Modulation Basics," May 2015. [Online]. Available: http://www.semtech.com/images/datasheet/an1200.22.pdf

[10] S. Corporation, *SX1301 datasheet*, 2014.

[11] N. Sornin, M. Luis, T. Eirich, T. Kramp, and O. Hersent, "LoRaWAN Specifications," LoRa Alliance, Tech. Rep., 2015.

[12] Matt Knight, "Reversing LoRa." [Online]. Available: https://github.com/matt-knight/research

[13] M. Knight and B. Seeber, "Decoding LoRa: Realizing a modern LPWAN with SDR," *Proceedings of the 6 th GNU Radio Conference*.

[14] Semtech Corporation, "Implementing Data Whitening and CRC Calculation in Software on SX12xx Devices." [Online]. Available: http://www.semtech.com/images/datasheet/AN1200.18_STD.pdf

[15] C. Goursaud and J.-M. Gorce, "Dedicated networks for IoT: PHY/MAC state of the art and challenges," *EAI endorsed transactions on Internet of Things*, 2015. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01231221

[16] M. Knight, "GnuRadio implementation of the LoRa modulation," 2016. [Online]. Available: https://github.com/BastilleResearch/gr-lora

[17] "Pothos implementation of a LoRa-like modulation," 2016. [Online]. Available: https://github.com/myriadrf/LoRa-SDR

[18] "The GNU Radio website," 2016. [Online]. Available: http://gnuradio.org/

[19] J. A. Gutierrez, E. H. Callaway, and R. Barrett, *IEEE 802.15.4 Low-Rate Wireless Personal Area Networks: Enabling Wireless Sensor Networks*. IEEE Standards Office, 2003.

[20] ETSI, "Electromagnetic compatibility and Radio spectrum Matters (ERM); Short Range Devices (SRD); Radio equipment to be used in the 25 MHz to 1000 MHz frequency range with power levels ranging up to 500 mW," Tech. Rep. EN 300 220-1 V2.4.1, Jan. 2012.

[21] CEPT, "ERC 70-03 Relating to the use of short range devices (SRD)," Tech. Rep., Sept. 2015.

[22] Actility, "ThingPark Wireless PHY and MAC layer specifications." [Online]. Available: http://cocoon.actility.com/system/files/wireless/ThingPark_Wireless%20PHY_and_MAC_layer_specifications_v1.pdf

[23] J. C. Ikuno, M. Wrulich, and M. Rupp, "System level simulation of LTE networks," in *IEEE Vehicular Technology Conference (VTC)*, May 2010.

[24] A. Augustin, J. Yi, T. Clausen, and W. M. Townsley, "A Study of LoRa: Long Range & Low Power Networks for the Internet of Things," *Sensors*, vol. 16, no. 9, Sept. 2016. [Online]. Available: http://www.mdpi.com/1424-8220/16/9/1466

[25] Semtech Corporation, "SX1272/3/6/7/8: LoRa Modem Designer's Guide AN1200.13," July 2013. [Online]. Available: https://www.semtech.com/images/datasheet/LoraDesignGuide_STD.pdf

[26] M. Bor, J. E. Vidler, and U. Roedig, "LoRa for the internet of things," 2016.

[27] J. Petajajarvi, K. Mikhaylov, A. Roivainen, T. Hanninen, and M. Pettissalo, "On the coverage of lpwans: range evaluation and channel attenuation model for lora technology," in *ITS Telecommunications (ITST), 2015 14th International Conference on*, Dec 2015.

[28] J. Petäjäjärvi, K. Mikhaylov, M. Hämäläinen, and J. Iinatti, "Evaluation of LoRa LPWAN technology for remote health and wellbeing monitoring," in *2016 10th International Symposium on Medical Information and Communication Technology (ISMICT)*, March 2016.

[29] M. C. Bor, U. Roedig, T. Voigt, and J. M. Alonso, "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*, ser. MSWiM '16, 2016.

[30] F. Adelantado, X. Vilajosana, P. Tuset-Peiró, B. Martínez, and J. Melià, "Understanding the limits of LoRaWAN," *CoRR*, vol. abs/1607.08011, 2016. [Online]. Available: http://arxiv.org/abs/1607.08011

[31] 3GPP, "Radio Frequency (RF) system scenarios," Tech. Rep. 36.942 V13.0.0, Jan. 2016.

[32] T. Petrić, M. Goessens, L. Nuaymi, A. Pelov, and L. Toutain, "Measurements, Performance and Analysis of LoRa FABIAN, a real-world implementation of LPWAN," 2016, working paper or preprint. [Online]. Available: https://hal-institut-mines-telecom.archives-ouvertes.fr/hal-01331966

[33] S. S. Szyszkowicz, H. Yanikomeroglu, and J. S. Thompson, "On the Feasibility of Wireless Shadowing Correlation Models," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 9, Nov. 2010.

[34] R. Fraile, J. F. Monserrat, J. Gozálvez, and N. Cardona, "Mobile radio bidimensional large-scale fading modelling with site-to-site cross-correlation," *European transactions on telecommunications*, vol. 19, no. 1, pp. 101–106, 2008.

[35] M. Gudmundson, "Correlation model for shadow fading in mobile radio systems," *Electronics Letters*, vol. 27, no. 23, pp. 2145–2146, Nov. 1991.

[36] H. Claussen, "Efficient modelling of channel maps with correlated shadow fading in mobile radio systems," in *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, vol. 1, Sept. 2005.

[37] S. Schlegel, N. Korn, and G. Scheuermann, "On the Interpolation of Data with Normally Distributed Uncertainty for Visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, Dec. 2012.

[38] P. L'Ecuyer, R. Simard, E. J. Chen, and W. D. Kelton, "An object-oriented random-number package with many long streams and substreams," *Oper. Res.*, vol. 50, no. 6, pp. 1073–1075, Nov. 2002. [Online]. Available: http://dx.doi.org/10.1287/opre.50.6.1073.358

[39] N. Benvenuto and M. Zorzi, *Principles of communications Networks and Systems*. Wiley Online Library, 2011.