Energy Saving Protocols for the Internet of Things

Thanh Hai To

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

Spécialité : Informatique

Arrêté ministériel : 25 mai 2016

Présentée par

Thanh Hai TO

Thèse dirigée par **Andrzej DUDA**, PR

préparée au sein du **Laboratoire Laboratoire d'Informatique de Grenoble**
dans **l'École Doctorale Mathématiques, Sciences et technologies de l'information, Informatique**

Protocoles à économie d'énergie pour l'Internet des Objets

Energy Saving Protocols for the Internet of Things

Thèse soutenue publiquement le **15 septembre 2020**,
devant le jury composé de :

Monsieur ANDRZEJ DUDA
PROFESSEUR DES UNIVERSITES, GRENOBLE INP, Directeur de thèse
Monsieur VIVIEN QUEMA
PROFESSEUR DES UNIVERSITES, GRENOBLE INP, Président
Monsieur CONGDUC PHAM
PROFESSEUR DES UNIVERSITES, UNIVERSITE DE PAU ET DES PAYS DE L'ADOUR, Rapporteur
Monsieur FABRICE VALOIS
PROFESSEUR DES UNIVERSITES, INSA LYON, Rapporteur

Résumé

Dans cette thèse, nous explorons des aspects performances de LoRa, l'une des technologies prometteuses pour des équipments IoT. LoRa appartient à une nouvelle classe de réseaux appelés "*Low Power Wide Area Networks*" (LPWAN) et définit une couche de radio spécifique basée sur la modulation "*Chirp Spread Spectrum*" et une méthode simple d'accès appelée LoRaWAN. Le fonctionnement de LoRaWAN est similaire à ALOHA: un appareil se réveille et envoie immédiatement un paquet à une passerelle. Ce choix de la méthode d'accès a un impact important sur la capacité de LoRa et son évolutivité vers un grand nombre d'appareils. Il en résulte un niveau élevé de paquets perdus suite aux collisions à mesure que le nombre d'appareils augmente.

Motivés par ces défis, nous avons conçu et mis en œuvre deux méthodes d'accès améliorées pour LoRa. Tout d'abord, nous avons proposé d'appliquer la technique "*Carrier Sense Multiple Access*" (CSMA) à LoRa pour diminuer le nombre de collisions. Nous avons utilisé le simulateur NS-3 pour évaluer le schéma et les résultats de la simulation montrent que CSMA réduit considérablement le taux de collisions tout en n'augmentant que légèrement la consommation d'énergie. Ensuite, nous avons proposé Timemaps, une nouvelle méthode d'accès pour améliorer les performances de LoRa. L'idée est de construire une carte temporelle de toutes les transmissions dans la passerelle et distribuer le planning de transmission lors de l'opération d'attachement d'un équipement au réseau. La carte temporelle permet un chevauchement des transmissions effectués avec des facteurs d'étalement différents. Comme des appareils allouent un facteur d'étalement en fonction de l'éloignement croissant de la passerelle, les transmissions deviennent orthogonales, ce qui conduit à une augmentation globale de la capacité du réseau.

Nous avons utilisé le simulateur NS-3 pour évaluer la méthode de Timemaps dans deux cas: des horloges parfaites et des horloges avec une dérive. La simulation prend en compte la quasi-orthogonalité des transmissions avec différents facteurs d'étalement et l'effet de capture. Les résultats montrent que Timemaps bénéficie d'un taux de livraison de paquets remarquablement plus élevé et du taux de collisions considérablement plus faible par rapport à LoRaWAN avec une consommation d'énergie modérément augmentée.

Pour valider les deux schémas proposés, nous avons développé un module NS-3

qui simule l'opération de LoRa à grain fin. Le module utilise un cadre énergétique implémenté dans NS-3 pour estimer la consommation d'énergie par nœuds et dans l'ensemble du réseau.

Abstract

In this dissertation, we consider performance aspects of LoRa, one of the promising technologies for lightweight smart sensors for the Internet of Things (IoT). LoRa belongs to a new class of Low Power Wide Area (LPWA) networks and defines a specific radio layer based on the Chirp Spread Spectrum modulation and a simple Medium Access Control called LoRaWAN. The operation of LoRaWAN is similar to ALOHA: a device wakes up and sends a packet to a Gateway right away. This choice of the access method highly impacts the capacity of LoRa and its scalability to a large number of devices. It results in a high level of packet losses due to collisions as the number of devices increases.

Motivated by these challenges, we have designed and implemented enhanced access methods to improve LoRa performance. First, we have taken advantage of Carrier Sense Multiple Access (CSMA) to lower the collision ratio. We have used the NS-3 simulator to evaluate the scheme and the simulation results show that CSMA considerably lowers the collision ratio while only slightly increasing energy consumption. Second, we have proposed Timemaps, a new access method for improving the performance of LoRa. The idea is to build a temporal map of all transmissions of IoT devices by a Gateway and distribute the schedule during join. Schedules admit overlapping transmissions of different spreading factors and as devices usually allocate a spreading factor in function of the increasing distance from the Gateway, transmissions will be in fact orthogonal, which leads to increased overall capacity of the network.

We have used the NS-3 simulator to evaluate Timemaps in both cases of perfect clocks and clocks with a drift. The simulation takes into account quasi-orthogonality of transmissions with different spreading factors and the capture effect. The results show that Timemaps benefits from remarkably higher packet delivery ratio and considerably lower the collision ratio compared to LoRaWAN along with moderately increased energy consumption.

To validate both schemes we have developed an NS-3 module that simulates the fine-grain LoRa operation. The module uses an energy framework implemented in NS-3 to estimate energy consumption at battery powered nodes and in the whole network.

Contents

Liste des figures

List of Tables

Glossary of Abbreviations

3GPP	Third Generation Partnership Project
8PSK	Eight-ary Phase Shift Keying
ABP	Activation By Personalization
ADR	Adaptive Data Rate
API	Application Program Interface
BER	Bit Error Rate
BPSK	Binary Phase Shift Keying
BW	Bandwidth
CAD	Channel Activity Detection
CAPEX	CAPital EXpenditure
CCG	Clear Channel Gap
CID	Command Identifier
CoAP	Constrained Application Protocol
CoMP-JT	Coordinated MultiPoint/Joint Transmission
CR	Coding Rate
CSMA	Carrier Sense Multiple Access
CSS	Chirp Spread Spectrum
DIFS	Distributed Coordinated Function Interframe Space
DIY	Do-It-Yourself
DRX	Discontinuous Reception
EC-GSM	Extended Coverage Global System for Mobile Communication

eDRX Extended Discontinuous Reception

eMTC Enhancements for Machine Type Communications

FDD Half Duplex Frequency Division Duplex

FEC Forward Error Correction

GMSK Gaussian Minimum Shift Keying

GPRS General Packet Radio Service

GPS Global Positioning System

GSM Mobile Communications

HSS Home Subscriber Server

IAB Internet Architecture Board

IoT Internet of Things

LPWA Low Power Wide Area

LTE Long Term Evolution

M2M Machine to machine

MAC Medium Access Control

MME Mobility Management Entity

MTC Machine Type Communication

MTU Maximum Transmission Unit

NB-IoT Narrow Band IoT

NSSE Network Synchronization and Scheduling Entity

OFDMA Orthogonal Frequency-Division Multiplex

OPEX OPerating EXpenses

OTAA Over The Air Activation

PCAP Packet CAPture

PER Packet Error Rate

P-GW Packet Data Network Gateway

PA Power Amplifier

PDR Packet Delivery Ratio

PRBs Physical Resource Blocks

PRR Packet Reception Rate

PSM Power Savings Management

RAN Radio Access Network

RF Radio Frequency

SC-FDMA Sub-Carrier Frequency-Division Multiplex

S-GW Serving Gateway

SF Spreading Factor

SIR Signal to Interference Ratio

SNR Signal to Noise Ratio

TBS Transport Block Size

TDMA Time Division Multiple Access

ToA Time on Air

TP Transmission Power

TTN The Things Network

UE User Equipment

UEs User Equipments

UNB Ultra-Narrow Band

Chapter 1

Introduction

Contents

1.1 Motivation

IoT corresponds to a vision of a world in which billions of devices with embedded intelligence, communication means, sensing and actuating capabilities will connect to the Internet. With the requirements of IoT applications such as long battery life, low cost, full coverage and support a massive number of devices, the Low Power Wide Area (LPWA) network is becoming an interesting candidate. The LPWA network includes the current proprietary LPWA technologies, such as LoRa, Sigfox, etc. and standardized IoT technologies of the Third Generation Partnership Project (3GPP). LPWA technologies are promising for the Internet of low power, low cost, and low throughput Things. A long range of LPWA technologies enables devices to operate over large geographical areas. IoT devices connected by LPWA technologies can be turned on anywhere and anytime to sense and interact with their environment instantly. It is worth noting that LPWA technologies achieve long range and low power operation at the expense of a low data rate. Nevertheless, the current status of the LPWA technologies, especially LoRa—an interesting technology for lightweight smart IoT sensing, raises open issues that can benefit from further improvements.

Indeed, LoRa defines a specific radio layer based on the Chirp Spread Spectrum (CSS) modulation and a simple channel access method called LoRaWAN [1]. The LoRa operation depends on a set of parameters: i) Bandwidth (BW), a range of spectrum for transmissions, Transmission Power (TP), iii) Spreading Factor (SF) that characterizes

the number of bits carried by a chirp to control the bit rate and reliability (higher SF means lower bit rate and lower BER, Bit Error Rate), and Coding Rate (CR) that defines the ratio of the redundant information for Forward Error Correction (FEC). The LoRa CSS modulation results in low sensitivity enabling transmissions over long distances: a range of several kilometers outdoors and hundreds of meters indoors. Depending on the duty cycle of LoRa devices, their lifetimes may become very long, for instance 17 years for a node sending 100 B once a day [2].

LoRaWAN [1] defines an access method to the radio channel similar to ALOHA: a device wakes up and sends a packet to the base station (Gateway in the LoRa terminology) right away. The difference with pure ALOHA is the variable packet length in LoRa. This choice of the access method highly impacts the capacity of LoRa and its scalability to a large number of devices. Several authors studied these aspects [3, 4, 5, 6, 7, 8, 9, 10, 11, 12]. The main conclusions to draw from the analyses are the following:

- the range of the LoRa network is limited to several kilometers, for instance: i) less than 10% of loss rate over a distance of 2 km for SF9-SF12 and ii) more than 60% of loss rate over 3.4 km for SF12. The coverage probability drops exponentially as the number of contending devices grows.
- the number of devices in a cell can be relatively large, but they are limited to sending a few bytes of data per day.
- the ETSI regulations of the 868 MHz ISM band set limits on the maximum duty cycle to 0.1% or 1% in the $863 - 870$ MHz ISM band (depending on the selected sub-band), which also limits the throughput of devices and the overall network capacity.
- the LoRaWAN operation similar to ALOHA results in a high level of packet losses due to collisions as the number of devices increases.
- the impact of collisions is significantly mitigated by the capture effect in which some transmissions that benefit from a stronger signal are successful despite of collisions.

Motivated by these challenges, we focus on the design and implementation of new access methods to address open issues affecting the performance of LoRaWAN. At the

same time, these access methods need to meet the energy consumption constraint of the devices in the LPWA network.

1.2 Overview of the Thesis

This thesis considers access methods in LPWA networks, especially LoRa, and provides three major contributions. First, we have developed a NS-3 module that simulates the behavior of LoRa. To validate the module, we have compared its results with measurements on both a real-world testbed and the measured values reported in other work [3]. The comparisons show a sufficient level of details to obtain meaningful results. Our second contribution targets a simple enhancement of LoRaWAN that slightly impacts energy consumption—the CSMA principle consisting of testing the channel if it is used by another transmission before attempting to send a packet [13]. We have used the NS-3 simulator of LoRa to evaluate CSMA compared to pure LoRaWAN. The third contribution is the schema called Timemaps for improving performance of LoRaWAN. The idea is to build a temporal map of all transmissions of IoT devices by a Gateway to schedule transmissions and avoid collisions.

The thesis is organized as follows:

Chapter 2 introduces IoT and LPWA networks. IoT concerns networks that interconnect "Things" able to autonomously exchange data. "Things" may be machines, parts of machines, smart meters, sensors, or even everyday objects such as retail goods or wearables. This capability will bring tremendous improvements in user experience and system efficiency. Communications enabled by IoT are referred to as Machine-Type Communication (MTC) characterized by the fact that devices exchange data without the need for human interaction. MTC concerns communication between devices and a server, or device-to-device, either directly or over a network. Concerning LPWA, we can categorize them into two separate sub-categories. On the one hand, there are proprietary LPWA technologies operating in the unlicensed bands of spectrum with the examples of SigFox and LoRa. On the other hand, there are the current and forthcoming 3GPP standardized cellular IoT technologies that typically operate in licensed spectrum bands.

Chapter 3 introduces the details of the LoRa physical layer, the LoRaWAN

medium access control (MAC) layer, and the network reference model for the LoRaWAN architecture. In addition, we discuss the adaptive data rate algorithm and the LoRaWAN performance.

Chapter 4 first introduces NS-3, an open source discrete-event network simulator, under the GNU GPLv2 license and is publicly available for networking research and education. Second, we show the details of our LoRa module in developed in NS-3. To validate the module, we have compared its results with measurements on both a real-world testbed and the measured values reported in other work [3].

Chapter 5 presents two main topics. First, we present the basic theory of ALOHA performance with two versions of pure ALOHA and slotted ALOHA. While the former achieves the maximum channel utilization of approximately 18 %, the latter obtains twice with 36 %. Second, we discuss CSMA, a method for improving the performance of ALOHA with its variants: Persistent and Nonpersistent CSMA. We have used the NS-3 simulator to evaluate CSMA in LoRa. The simulation results show that CSMA considerably lowers the collision ratio while only slightly increasing energy consumption.

Chapter 6 presents Timemaps, a new access method for improving the performance of LoRa. The idea is to build a temporal map of all transmissions of IoT devices by a Gateway and distribute the schedule during join. Schedules admit overlapping transmissions of different SF and as devices usually allocate SF in function of the increasing distance from the Gateway, transmissions will be in fact orthogonal, which leads to increased overall capacity of the network. We have used the NS-3 simulator to evaluate our proposal in both cases of perfect clocks and clocks with a drift. The simulation takes into account quasi-orthogonality of transmissions with different spreading factors and the capture effect. The results show that Timemaps benefits from remarkably higher Packet Delivery Ratio (PDR) and considerably lower the collision ratio compared to LoRaWAN along with moderately increased energy consumption.

Chapter 7 terminates the thesis by summarizing the main contributions and outlining further research directions.

Chapter 2

Internet of Things and Low Power Wide Area Networks

Contents

2.1 Introduction

In this chapter, we present an overview of the Internet of Things (IoT) and Low Power Wide Area (LPWA) networks.

2.2 Internet of Things

2.2.1 Introduction

IoT is the next revolution in the mobile ecosystem. IoT services seem to be the main driver for the further growth of mobile networks. It is predicted that there will be about 30 billion connected devices deployed by 2025, of which mobile IoT technology (i.e., 2G, 3G, and 4G technologies used for IoT but not specifically optimized for IoT) and LPWA modules are predicted to account for about 7 billion in 2025 [14].

The idea of connecting objects to each other and to the Internet is not new. However, why is IoT a newly popular topic today? From a broad perspective, the confluence of several technology and market trends is making it possible to interconnect more and smaller devices cheaply and easily [15]. We discuss these aspects below:

- Ubiquitous connectivity: supports communication for everyone and everything, anywhere, anytime, thanks to its low cost, high speed, ubiquitous networking, especially through licensed and unlicensed wireless technologies and services.

- Advances in data science: new algorithms, along with a rapid increase in computing power, data storage capabilities and cloud services allow the aggregation, correlation, and analysis of large amounts of data. These large datasets provide new opportunities to extract information and knowledge.

- Extensive adaptation of IP-based networks: The IP protocol at the foundation of the Internet has become a common standard globally. The suite of Internet protocols provides a complete and useful software platform and tools, which can be easily integrated into most devices.

- Miniaturizing IoT design: advances of the hardware industry allow computing and communications technology to be integrated into very small objects. Ideally, engineers want to use the smallest IoT components possible, with excellent Radio Frequency (RF) performance and affordable prices. These features often do not meet in IoT component services, and that indicates a challenge for solution providers. However, the size of a silicon integrated circuit has become smaller and smaller over the years as the industry adopted new silicon manufacturing processes. This has driven the advancement of small and inexpensive sensor devices, enabling many IoT applications.

- Rise of Cloud Computing: Cloud Computing allows small and distributed devices to interact easily, along with powerful analytical and control support in the backend infrastructure. This is thanks to Cloud Computing ability to leverage remote-connected computing resources to process, manage, and store data.

2.2.2 IoT Market Landscape

IoT concerns Machine Type Communications (MTC) that do not involve human interaction. The total MTC market is expected to attain 30 billion devices by 2025. Of which, fixed and short range communication will be used for most connections. However, there are also a significant number (about seven billion by 2025) of predicted connections via traditional cellular IoT and LPWA networks [14].

2.2.3 IoT Application Requirements

To successfully support massive MTC deployment, IoT applications need to address the following key requirements [14]:

- Long battery life
- Low cost
- Full coverage
- Massive number of devices

Long battery life: many IoT devices have to operate for very long times, usually years. For example, with LoRa devices, depending on the duty cycle, their lifetimes may become very long, for instance 17 years for a node sending 100 B once a day [2].

Low cost: IoT devices and their connectivity are cheaper than the legacy networks. The industry objective is achieving a module cost that less than 5 USD. To make possible a positive case for cellular IoT in business, the overall cost of ownership involving the device must be very low.

To obtain a low deployment cost, the cost of IoT connectivity, comprising initial CAPital EXpenditure (CAPEX) and annual OPerating EXpenses (OPEX), have to be kept to a minimum. Deploying IoT connectivity on topmost of existing cellular networks can be reached by software upgrade so as to avoid any new hardware, keeping CAPEX and OPEX to a minimum level.

One of the examples of low cost solutions is the WAZIUP IoT platform proposed by Pham et al. [16, 17]. It is a low cost infrastructure for deploying IoT in developing

countries. To keep down the costs of sensing and the infrastructure, they adopted an approach of Do-It-Yourself (DIY) that integrates only low power and low cost off-the-shelf components. This open platform is currently deployed in three testbeds, dealing with soil monitoring in Senegal, water quality in fish ponds in Ghana, waste management and collection in an urban area in Togo. They aim to provide a sustainable solution for Western African countries by considering not only the environmental component but also the local economic and social aspects of this region.

Full coverage: enhanced coverage is important to many IoT applications. A simple example a smart meter, placed usually in the basement of buildings, behind concrete walls. The goal for the IoT connectivity link budget is an enhancement of 15-20 dB. Reinforcing coverage may allow deeper indoor connectivity through better penetratinon of walls or floors.

Massive number of devices: the number of IoT connected devices is growing significantly and it is estimated that by 2025 there will be seven billion devices connected via mobile IoT networks. This is equivalent to the current number of global mobile subscriptions. Because the density of connected IoT devices may not be identical in each place, some cells may have a higher number of connected devices than others.

2.2.4 IoT Communication Models

In this section, we present the model of IoT device connection and communication in the smart object environment. In March 2015, the Internet Architecture Board (IAB) released an architectural document for networking of smart objects [18]. The document defines a framework of four common communication models used by IoT devices.

2.2.4.1 Device-To-Device Communication Model

The first model is the device-to-device communication model. Figure 2.1 presents an example of this model in which two devices developed by different manufacturers want to interoperate and communicate directly. They may communicate over many types of networks but usually they use energy effficient short-range protocols like Bluetooth, Z-Wave, or ZigBee to set up direct device-to-device communications [15]. In addition,

Figure 2.1 – Example of a Device-To-Device Communication Model.

to fulfill the promise that devices from different manufacturers are able to communicate out of the box, these vendors need to agree on the protocol stack.

The device-to-device communication model is often used in applications such as home automation systems, which typically use small data packets to communicate between devices with low data rates. For example, with a home automation application based on Bluetooth technology running on a mobile phone, by sending its commands to appliances in our house through Bluetooth communication, we can monitor and control the appliances such as turning on or off the lights, the radiators, etc.

2.2.4.2 Device-To-Cloud Communication Model

In the Device-to-Cloud communication model, an IoT device connects to an Internet cloud service or an application service provider to exchange data and control traffic. Figure 2.2 shows an example of uploading sensor data to an application service provider [18]. This approach frequently takes advantage of existing communication technologies like 802.11 or BLE to set up a connection between the device and the IP network, which finally provides connectivity with the cloud service.

Although this model allows the use of IP-based end-to-end communication, it can still lead to vertical silos (i.e., any management system that is unable to operate with any other system, meaning that it is closed off from other systems). To prevent silos, service providers may allow third-party device vendors to connect to their server infrastructure. For those cases, the protocol interface used to communicate with the server infrastructure needs to be provided and available with different standards, such as Constrained Application Protocol (CoAP) or HTTP as shown in Figure 2.2.

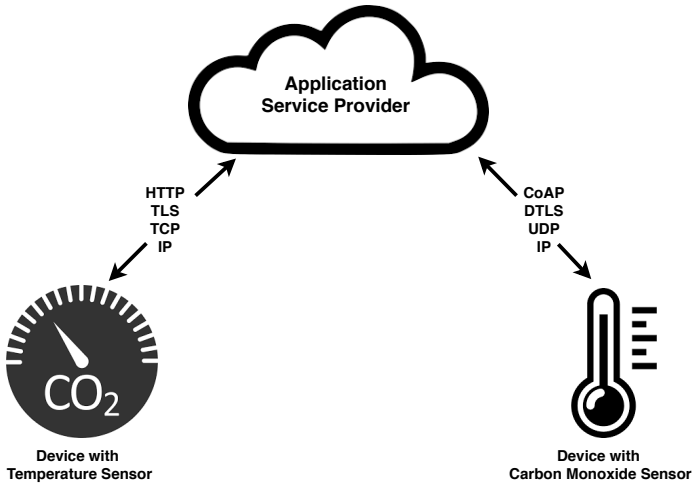

Figure 2.2 – Example of a Device-To-Cloud Communication Model.

2.2.4.3 Device-to-Gateway Communication Model

The Device-to-Cloud communication model described in Section 2.2.4.2 works well if it uses a radio technology widely deployed in the targeted market, such as IEEE 802.11 for smart home use cases. Sometimes, other radio technologies are needed (such as IEEE 802.15.4) or a specific application-layer functionality (e.g., local authentication and authorization) has to be provided or interoperability is needed with legacy, non-IP-based devices. In those cases, some form of a gateway has to be introduced into the communication architecture that bridges between different technologies and supports other networking and security functionality. Figure 2.3 illustrates this model.

In the Device-to-Gateway communication model, IoT devices connect via a gateway to cloud services. The gateway acts as an intermediate device to forward data exchanged between the devices and cloud services.

Devices send packets to gateways using different standards such as CoAP or HTTP. The gateways, in turn, connects to the cloud services over a high throughput backhaul network (cellular, Ethernet, or satellite). The cloud services provide the management of gateways and devices.

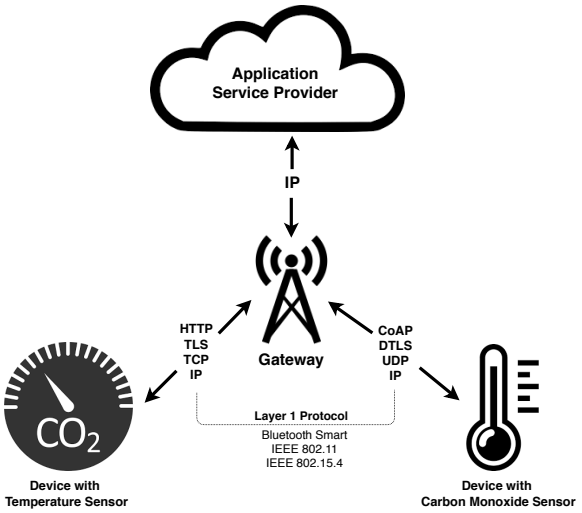

Figure 2.3 – Example of a Device-To-Gateway Communication Model.

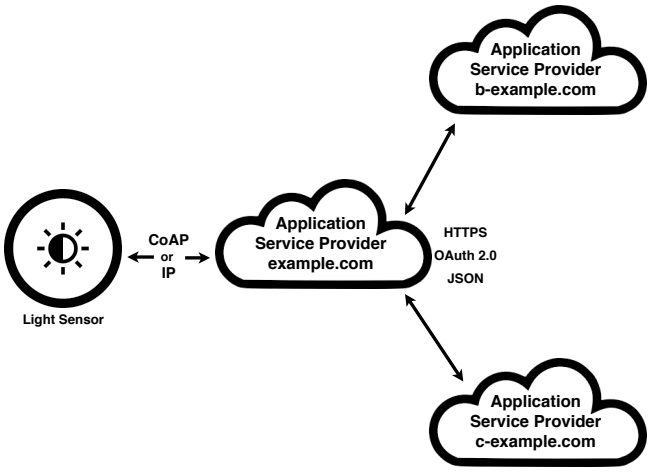

Figure 2.4 – Example of a Back-End Data-Sharing Communication Model.

2.2.4.4 Back-End Data-Sharing Communication Model

In the back-end data-sharing communication model, users extract and analyze data from cloud services in conjunction with data from other sources. This model is an extension of the single device-to-cloud communication model, which can lead to silos phenomena in which IoT devices only upload data to a single application service provider. The back-end data-sharing communication model also allows the data collected from the single data stream of an IoT device to be aggregated and analyzed. Figure 2.4 illustrates this model.

This model comes from the Web and is reapplied to the smart object context: typically, it is based on a RESTful Application Program Interface (API) together with

the reuse of a federated authentication and authorization technology (like OAuth 2.0—an authorization framework that enables a third-party application to obtain limited access to an HTTP service [19])

2.3 Low Power Wide Area Networks

LPWA networks have become a popular radio communication technology because of their main characteristics: large coverage areas and low energy consumption [20].

They represent a novel communication paradigm that complements cellular and wireless technologies for short range in addressing IoT applications with diverse requirements. LPWA technologies provide a set of features including long range connectivity for low data rate and low power devices. Their market is expected to be very large and approximate $\frac{1}{4}$ of overall 30 billion IoT devices are able to be connected to the Internet using LPWA technologies, either based on cellular technologies or proprietary [14]. There are certain sectors that can take advantage of LPWA technologies to connect their devices such as health monitoring [21], agriculture monitoring [22, 17], traffic monitoring [23], localization [24], smart transport system [25, 26, 27], smart city [28], and smart grid [29].

In general, traditional cellular technologies are not able to achieve energy efficiency in the battery for ten years. The cost and complexity of devices in cellular networks is high because of its capacity to address the optimization for voice, high speed data services and complex waveforms. For low power IoT devices, there is a clear need to strip complexity to reduce cost [20]. Efforts in this direction are underway for cellular networks by 3GPP and are covered in detail in Section 2.3.3.

LPWA technologies allow devices to be able to work over big geographical areas. They are, however, not possible to address all IoT use cases. LPWA technologies are taken into account for those use cases that require low power, low cost and do not need high data rates. Such applications are categorized as Massive MTC applications which respond to the requirements of many applications for health monitoring, agriculture monitoring, traffic monitoring, smart city, smart grid, etc. that exchange a small amount of data infrequently. Therefore, even though limited by low data rate, the attraction of LPWA technologies continue to expand, typically LoRa and Sigfox. The

details of LoRa and Sigfox are presented in Sections 2.3.1 and 2.3.2, respectively.

2.3.1 LoRa and LoRaWAN

LoRa [30] belongs to the new class of LPWA networks and defines a specific radio layer based on the Chirp Spread Spectrum modulation and a simple channel access method called LoRaWAN [1]. We present the details of the LoRa technology in Chapter 3.

2.3.2 Sigfox

Sigfox offers an end-to-end LPWA connectivity solution based on its patented technologies. The network operators of this technology deploy the proprietary base stations equipped with cognitive software-defined radios and connect them to the backend servers using an IP-based network. End devices communicate with the base stations using Binary Phase Shift Keying (BPSK) modulation in an Ultra Narrow Band (UNB) of 100 Hz [20].

Sigfox provides a way of collecting data from sensors and devices with a set of API. Besides, it complements traditional cellular Machine to Machine (M2M) by enabling global, ubiquitous, long battery life solutions at low cost [31].

Sigfox use a 192 kHz of the publicly available band with a data rate of 100 or 600 bits per second depending on the region, which enables communication over long distances without being impacted by the noise.

The used band depends on the location. In Europe, for example, the used band is between 868 and 868.2 MHz; in the rest of the world, it is between 902 and 928 MHz with restrictions defined by local regulations [31].

A Sigfox device transmits a message on a random frequency and then sends 2 replicas on different frequencies and time, called “time and frequency diversity” presented in the Figure 2.5. A message with a 12-byte payload takes 2.08 s over the air with a rate of 100 bps. The Sigfox base stations monitor the full 192 kHz spectrum and look for UNB signals to demodulate.

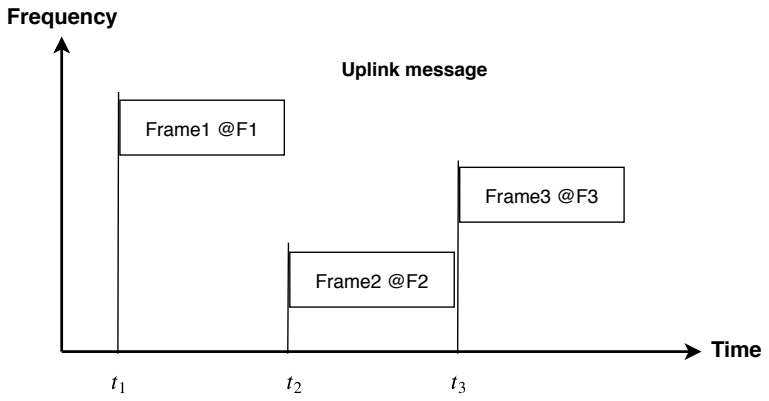

Figure 2.5 – Sigfox frequency hopping on replicas.

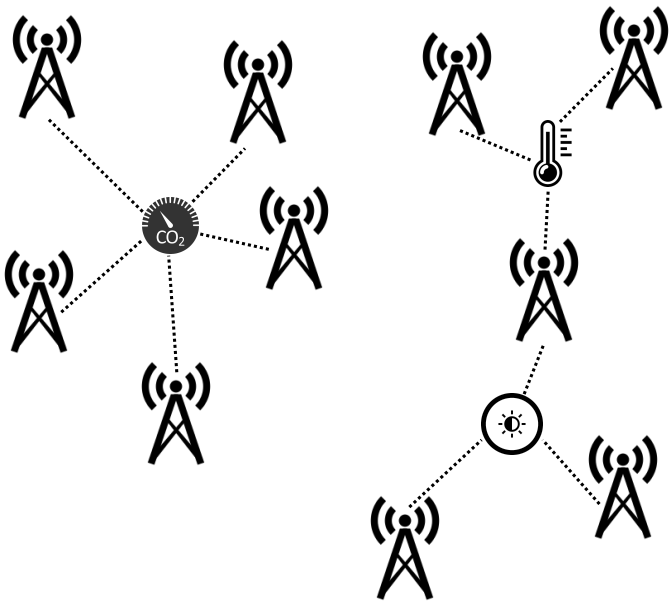

Figure 2.6 – Message reception by multiple Sigfox base stations.

The principle of the cooperative reception consists of the possibility of receiving a message by several base stations, because a device is not attached to a specific base station unlike cellular protocols. The emitted message is received by any base stations that are nearby and on average, the number of base stations is 3 (see Figure 2.6).

To deal with the autonomy constraints and cost of devices, Sigfox has developed a communication protocol for small messages. The size of the message is in the range between 0 and 12 bytes. Some payload size examples are listed below [31]:

- GPS (Global Positioning System) coordinates: 6 bytes
- Temperature: 2 bytes

- Speed reporting: 1 byte
- Object status: 1 byte
- "Keep alive" message: 0 byte

Sigfox supports bidirectional communication. The downlink communication can only precede uplink communication after which the device should wait for a response from the base station. The regulation in Europe states that devices can occupy most of the channels in the public 868 MHz ISM band for 1% of the time, which translates into 6 12-byte messages per hour or 140 messages per day. The radio access link of Sigfox is asymmetric. For the downlink messages from the base stations to devices, it permits a maximum of transmission at 4 8-bytes messages per day. In addition, the duty cycle configured for the base station is 10% to ensure that there are 4 downlink messages per day per device. If there are additional resources left, the device can receive more.

2.3.3 3GPP Cellular Solutions for the Internet of Things

To deal with IoT and M2M markets, 3GPP is developing its existing cellular standards to reduce cost and complexity, improve the signal penetration, range, and extend the battery lifetime. Its solutions such as 4G Long Term Evolution (LTE) enhancements for Machine Type Communications (eMTC), Extended Coverage Global System for Mobile Communication (EC-GSM), and Narrow Band IoT (NB-IoT) offer different trade-offs between cost, coverage, data rate, and power consumption to come up with the diverse requirements of M2M and IoT applications. However, the general objective of these standards is to reuse the existing cellular infrastructure and owned radio spectrum as much as possible [20]. In this section, we introduce these 3GPP improved cellular solutions for IoT.

2.3.3.1 eMTC

The LTE Enhancements for Machine Type Communications (eMTC), also called LTE Cat-M1, or Cat-M, is a cellular LPWA technology introduced in the 3GPP Release-13 standardization, which intends to minimize modem complexity and cost, power con-

sumption, and extended coverage over existing legacy handset modems, such as category 0 User Equipments (UEs) from Release-12 specification for MTC. This technology is an enhancement for LTE networks to support MTC for IoT.

Cat-M1 User Equipment (UE) operates within a limited bandwidth of 1.08 MHz out of the available 1.4 MHz. Because of this, it only uses six Physical Resource Blocks (PRBs) out of the eight available 180 kHz LTE PRBs, which coexist in a broader, general legacy-purpose LTE system. To mitigate the interference level, the two remaining PRBs are used as guard bands. With the support for 1.08 MHz band (narrowband channel) for both radio frequency and baseband, Cat-M1 devices are further reduced in complexity, cost and power over Cat-0 devices introduced in the 3GPP Release-12 standardization. Cat-M1 devices are expected to achieve a maximum throughput of up to 1 Mbps in both uplink and downlink operations for massive IoT. For common control messages, the maximum Transport Block Size (TBS) is further reduced to 1000 bits from the 2216 bits of Cat-0 devices which is an equivalent of unicast data traffic, allowing further processing and memory savings in Cat-M1 devices over the legacy Cat-0 UE [32].

The eMTC devices have been designed to support either 23 dBm or 20 dBm power classes unlike the MTC Cat-0 devices, which were designed to support a maximum transmission power of 23 dBm, which is approximately 200 mW for uplink. The maximum transmission power of 20 dBm enables the Power Amplifier (PA) to be integrated as opposed to using a dedicated PA. Consequently, this enables and supports the achievement of a lower device cost.

eMTC uses Power Savings Management (PSM) and Extended Discontinuous Reception (eDRX) to achieve long battery life for Cat-M1 devices. Thanks to these power savings mechanisms, the design objective for service life of terminal battery for eMTC in 3GPP standards is 10 years [33]. For coverage, the design objective is to obtain coverage enhancement of 15 dB compared to LTE. If the maximal coupling path loss of LTE is 144 dB, that of eMTC should be 155 dB.

2.3.3.2 EC-GSM

3GPP proposed the Extended Coverage GSM (EC-GSM) standard that aims to extend the Global System for Mobile Communications (GSM) coverage by +20 dB using

sub GHz band in the context of indoor networks for better signal penetration. A link budget of this IoT solution is in the range of 154 to 164 dB depending on the transmission power. With just a software upgrade of GSM networks, the legacy General Packet Radio Service (GPRS) spectrum is able to fill up the new logical channels to hold EC-GSM devices. EC-GSM benefits signal processing techniques and repetitive transmissions to improve capacity and coverage of legacy GPRS. It provides variable data rates based on two modulation techniques namely Eight-ary Phase Shift Keying (8PSK) and Gaussian Minimum Shift Keying (GMSK). This standard was released in the middle of 2016. It intends to support up to 50 000 devices per base station. It also aims to improve security and privacy in comparison to solutions based on conventional GSM [20]. In addition, the battery lifetime of each node in EC-GSM is about 10 years with a battery of a 5 Wh, relying on some factors such as the power class used, the number of bytes required to send per day, the distance between the device and the base station. For example, one device utilizing the class of 33 dBm power, giving a coverage of 154 dB, and transmitting 50 bytes during 2 hours is able to be expected to extend over 14 years (see Tables [6.2.6.6-9] - [6.2.6.6-12] in [34]).

2.3.3.3 NB-IoT

The 3GPP introduced the first IoT-specific UE in LTE Release 12, known as LTE Cat-0 or LTE-M. LTE-M features include peak data rate of 1 Mb/s over 1.08 MHz bandwidth and support for UEs with half duplex operation and power saving mode. In its LTE Release 13, 3GPP has standardized a new Radio Access Network (RAN) technology called Narrow Band IoT (NB-IoT). Essentially, NB-IoT has been designed for low cost, long battery life, high coverage, and deployment of a large number of devices. It inherits basic functionalities from the LTE system, while it operates in a narrow band. With a software upgrade, core network elements of an operator existing LTE network can be enabled to support NB-IoT, which is essential for reducing deployment cost and time [35].

NB-IoT offers several benefits such as less complexity in transceiver design, low power consumption, lower cost for the radio chip, and coverage enhancement. Discontinuous Reception (DRX) allows devices to save power by going to sleep, and occasionally waking up and listening for incoming data and paging messages from the network. The periodicity by which devices should wake up is configured by DRX cycles. NB-

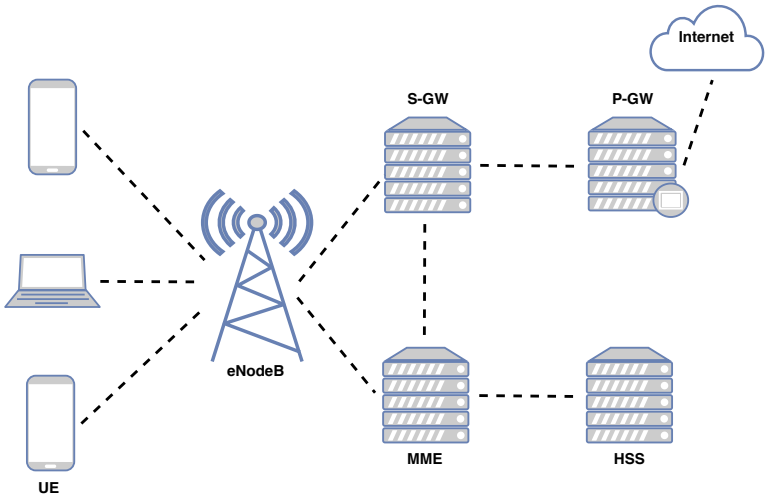

Figure 2.7 – The 3GPP network architecture.

IoT supports extended DRX (eDRX) cycles, which can reach up to 10 s for UEs in connected state and about 3 h for UEs in idle state. Due to its small bandwidth, the power spectral density of NB-IoT carrier can be boosted by 6 dB compared to legacy LTE. Moreover, the received signal quality can be improved by combining multiple repetitions. As a result, NB-IoT offers 20 dB coverage enhancement compared to LTE. Hence, if the maximal coupling path loss of LTE is 144 dB, the maximal coupling path loss of NB-IoT should be 164 dB. In addition, NB-IoT aims to support long battery life. For a device with 164 dB coupling loss, a battery life of 10 years can be achieved if the UE transmits 200-byte data a day on average [34].

Figure 2.7 shows the 3GPP network architecture that applies to NB-IoT [36]. It includes the following components:

- UE is User Equipment or User Device.
- eNodeB is a component connected to the mobile phone network and communicates directly with UEs.
- Mobility Management Entity (MME) is in charge of controlling the mobility of UEs. It tracks UEs, manages session, chooses the Serving Gateway for UEs during the process of initial attachment and user authenticating.
- The Serving Gateway (S-GW) is responsible for routing and forwarding data packets through the network. It works as an anchor for UEs in the handover between eNodeBs and the handovers between NB-IoT and other 3GPP solutions.

- The Packet Data Network Gateway (P-GW) acts as an interface between external networks and the 3GPP networks.
- The Home Subscriber Server (HSS) holds information about users and subscriptions. It provides a database that allows us to manage mobility, support session-establishment, authenticate user, and authorize access.

NB-IoT supports Half Duplex Frequency Division Duplex (FDD) operation mode with a Maximum Transmission Unit (MTU) size of 1600 bytes. Any packet with the size up to the size of MTU can be goes through the NB-IoT stack from higher layers. NB-IoT utilizes narrowbands with a bandwidth of 180 kHz in both uplink and downlink. The access method used in the downlink of NB-IoT is Orthogonal Frequency-Division Multiplex (OFDMA). It uses the 15 kHz sub-carrier spacing. With uplink, Sub-Carrier Frequency-Division Multiplex (SC-FDMA) single tone is used together with the 15 kHz or 3.75 kHz tone spacing.

NB-IoT provides three ways of deployment. First, with the *in-band deployment*, the narrowband is deployed inside the LTE band and radio resources are shared flexibly between normal LTE carrier and NB-IoT. Second, with the *in-guard-band deployment*, the narrowband utilizes the unused resource blocks between two contiguous LTE carriers. Third, with the *standalone deployment*, the narrowband can be situated alone in the dedicated spectrum that makes it able to reframe a GSM 850/900 MHz for NB-IoT. These deployment modes are applied in licensed bands. The maximum transmission power for uplink is 20 or 23 dBm, while that for downlink is higher, up to 46 dBm depending on the specific deployment [36].

Table 2.1 provides a summary of NB-IoT, LoRa, and Sigfox in terms of key features [32, 34, 37, 38].

Table 2.1 – Comparison between NB-IoT, LoRa and Sigfox.

Technical metrics	**NB-IoT**	**LoRa**	**Sigfox**
Modulation	QPSK	CSS	BPSK
Frequency	Licensed LTE bands	Unlicensed ISM bands	Unlicensed ISM bands
Bandwidth	180 kHz	125 kHz, 250 kHz or 500 kHz	100 Hz
Bidirectional	Yes/Half-duplex	Yes/Half-duplex	Limited/Half-duplex
MAC layer	LTE based	ALOHA based	ALOHA based
Authentication & encryption	Yes (LTE encryption)	Yes (AES 128b)	Not supported
Adaptive data rate	No	Yes	No
Maximum data rate	50 kbps	50 kbps	100 bps
Coverage	<25 km	<20 km	<40 km
Power consumption (depends on traffic/ deployment)	>10 years [34] battery life	>10 years battery life	>10 years battery life
Duty-Cycle/LBT restriction	No	0.1-1% or LBT (depends on region)	No
Allow private network	No	Yes	No

2.4 Conclusion

This chapter discusses two aspects. First, we present an overview of IoT, its market landscape, IoT application requirements, and its communication models. Second, we present LPWA networks with the emphasis on the technical differences of Sigfox, LoRa, and 3GPP improved cellular solutions for IoT. Each technology has its place in the IoT market landscape. LoRa and Sigfox are suitable for low-cost deployments with extended coverage, and long battery lifetimes. In contrast, NB-IoT fits into the role of IoT applications that require very low latency and high quality of service.

Chapter 3

LoRa and LoRaWAN

Contents

3.1 Introduction

LoRa has become an interesting technology for lightweight smart IoT sensing [30]. It belongs to the new class of LPWA networks and defines a specific radio layer based on the Chirp Spread Spectrum (CSS) modulation and a simple channel access method called LoRaWAN [1]. In this chapter, we introduce the LoRa physical layer, the LoRaWAN channel access method, the network reference model for LoRaWAN architecture, the adaptive data rate algorithm, and discuss performance issues.

3.2 LoRa Physical Layer

The LoRa physical layer is based on the CSS modulation that provides low sensitivity needed for long communication ranges [39]. Communication between devices and

gateways can take place simultaneously on multiple frequency channels. Each device can transmit its packet with a specific SF. Higher values of SF result in longer communication ranges. Typical values of bandwidth (BW) are 125, 250, and 500 kHz in the ISM 868 and 915 MHz bands [11]. Sensitivity ranges from -136 dBm for SF12 and BW of 125 kHz to -111 dBm for SF6 and BW of 500 kHz. Coding Rate (CR) can be $\frac{4}{5}$, $\frac{4}{6}$, $\frac{4}{7}$, or $\frac{4}{8}$. For instance, for BW of 125 kHz and CR of $\frac{4}{5}$, the bit rate is 5468 b/s with SF6 and 293 b/s with SF12.

3.2.1 LoRa Spread Spectrum

LoRa modulation is based on CSS to provide a low cost, low power, and robust alternative to the traditional spread spectrum communication techniques. In this modulation, the spreading of the spectrum is reached by rendering a chirp signal that continuously varies in frequency. Because of this, timing and frequency offsets between transmitter and receiver are identical, leading to reduce significantly the complexity of the receiver.

To further improve the robustness against interference, the LoRa modulation includes a variable error correction scheme. The symbol rate and data rate is determined based on the bandwidth and the SF used. The symbol rate, R_s is defined as:

$$R_s = SF * \frac{BW}{2^{SF}} \tag{3.1}$$

and, the data rate, R_b is defined as:

$$R_b = SF * \frac{\left[\frac{4}{4+CR}\right]}{\left[\frac{2^{SF}}{BW}\right]} bits/sec \tag{3.2}$$

where:

- SF is spreading factor (7..12)
- CR is code rate (1..4)
- BW is modulation bandwidth (Hz)

Table 3.1 – Semtech SX1276 LoRa.

BW \ SF	7	8	9	10	11	12
125 kHz	-123	-126	-129	-132	-133	-136
250 kHz	-120	-123	-125	-128	-130	-133
500 kHz	-116	-119	-122	-125	-128	-130

Table 3.2 – LoRa parameters for BW of 125 kHz.

SF	Chirps/ symbol	SNR limit	Airtime	Bit rate	PL_{max}
7	128	-7.5 dB	102.7 ms	DR5: 5469 b/s	230 B
8	256	-10 dB	184.8 ms	DR4: 3125 b/s	230 B
9	512	-12.5 dB	328.7 ms	DR3: 1758 b/s	123 B
10	1024	-15 dB	616.5 ms	DR2: 977 b/s	59 B
11	2048	-17.5 dB	1315 ms	DR1: 537 b/s	59 B
12	4096	-20 dB	2466 ms	DR0: 293 b/s	59 B

These parameters (BW, SF, and CR) also influence the sensitivity of the decoder. In general, an increment of BW lowers the receiver sensitivity, whereas an increment of the SF increases the receiver sensitivity. Decreasing CR leads to the reduction of the Packet Error Rate (PER) in the presence of interference. Table 3.1 taken from the SX1276 datasheet [40] presents the receiver sensitivity in [dB] at different BWs and SFs.

Table 3.2 presents spreading factors SF$\{12 - j\}$, which result in data rates DRj, with $j = 0, 1, \ldots$ 5, the SNR limit, as well as the maximal payload length PL_{max} and its airtime for the bandwidth of 125 kHz.

3.3 LoRaWAN

This section introduces classes of devices in LoRaWAN, formats of physical and MAC messages, list of MAC commands, and the process of device activation in the LoRa network.

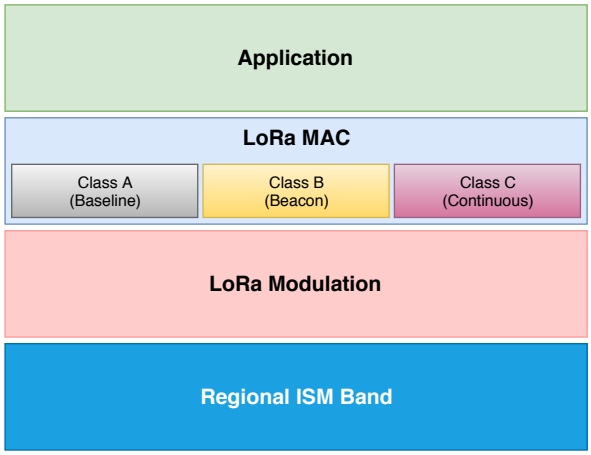

Figure 3.1 – LoRaWAN classes.

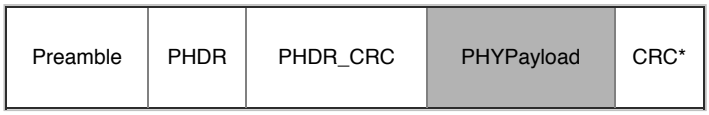

Figure 3.2 – Radio PHY structure of messages (Cyclic Redundancy Check* field is only available in uplink transmissions).

3.3.1 LoRaWAN Classes

LoRaWAN defines three types of devices, namely *Class A*, *B*, and *C* as in Figure 3.1. *Class A* devices use an ALOHA protocol for the uplink. After sending a packet, a device listens to a response from the gateway during two downlink receive windows. *Class A* results in the lowest energy consumption, so we only consider this class in the thesis. *Class B* devices aim at applications requiring more downlink traffic. The devices open extra receive windows at scheduled times by receiving a time-synchronized beacon from the gateway. *Class C* devices are always on and listen to the channel all the time, so their energy consumption is the highest. Only *Class A* must be implemented in all end devices.

3.3.2 Physical Message Formats

Uplink and downlink messages of LoRa hold a PHY payload ($PHYPayload$) as illustrated in Figure 3.2. It begins with a MAC header ($MHDR$, single-octet), followed by a MAC payload ($MACPayload$), and terminating with a message integrity code (MIC, 4-octet). We can see these components in the first part of Figure 3.3.

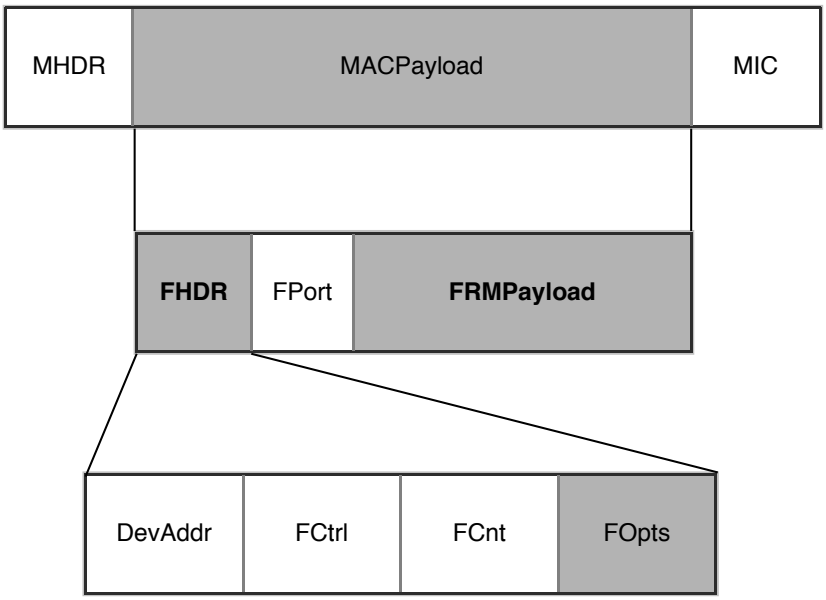

Figure 3.3 – LoRaWAN Medium Access Control message format. $FHDR$ is 7 bytes if it contains no frame options, and up to 22 bytes when the frame options are used. The maximum size of frame options, $FOpts$, is 15 bytes. A data frame can hold any sequence of MAC commands, either piggybacked in the $FOpts$ field if the $FPort$ field being set to greater than 0 or, when sent as a separate data frame, in the $FRMPayload$ field if the $FPort$ field being set to 0.

Devices sent uplink messages to a Network Server. These messages relayed by one or many gateways. The Network Server sends one downlink message to only one device relayed by a single gateway.

3.3.3 MAC Message Formats

The MAC payload of data messages contains a frame header ($FHDR$) followed by an optional port field ($FPort$) and an optional frame payload field ($FRMPayload$) as in Figure 3.3. The $FHDR$ contains the short device address of the device ($DevAddr$), a frame control octet ($FCtrl$), a 2-octet frame counter ($FCnt$), and up to 15 octets of frame options ($FOpts$) used to transport MAC commands [1].

The MAC payload of the data messages can carry MAC commands in $FHDR$ or $FRMPayload$, depending on the $FPort$ value. While the maximum value of $FHDR$ is 15 bytes, that of $FRMPayload$ according to data rates are shown in Table 3.3. Therefore, if the sequence of MAC commands is small, the $FHDR$ field will be used.

Otherwise, if the sequence is large, the device uses *FRMPayload* to carry MAC commands.

Table 3.3 – LoRaWAN maximum payload sizes

Data rate (DR)	SF	Bandwidth (kHz)	Maximum MACPayload size (bytes)	Maximum FRMPayload size (bytes)
1	12	125	59	51
2	11	125	59	51
3	10	125	59	51
4	9	125	123	115
5	8	125	250	242
6	7	125	250	242
7	7	250	250	242

3.3.4 MAC Commands

For network administration and management, a number of MAC commands can be exchanged between the Network Server and devices. The list of all the LoRaWAN MAC commands in LoRaWAN specification 1.1 is shown in Tables 3.4 and 3.5. It consists of a Command Identifier (CID), Command Name, transmitted by Device or Gateway, and a Brief Description about the purpose of the command.

A data frame can contain any series of MAC commands. It is either piggybacked on the *FOpts* field when sent as a separate data frame or in the *FRMPayload* field with the *FPort* field being set to 0 (see Figure 3.3). One series of the MAC commands is sent encrypted and not greater than 15 octets. A sequence of MAC commands is answered by the receiving end in the same order that they are transmitted.

3.3.5 Receive Windows

Following each uplink transmission, the device opens two short receive windows. Their start times are defined using the end of the transmission as a reference. Figure 3.4 shows the receive slot timing.

Table 3.4 – List of MAC commands

CID	Command Name	Device	Gateway	Description
0x01	ResetInd	x		The ABP device uses this to indicate a reset to the network and negotiate the protocol version
0x01	ResetConf		x	Answers to the ResetInd command
0x02	LinkCheckReq	x		A device use this to validate its connectivity to a network
0x02	LinkCheckAns		x	Answers to the LinkCheckReq command
0x03	LinkADRReq		x	Network Server requests a device to perform a rate adaptation
0x03	LinkADRAns	x		Answers to the LinkADRReq command
0x04	DutyCycleReq		x	The network coordinator sets the maximum aggregated transmit duty cycle of a device
0x04	DutyCycleAns	x		Answers to the DutyCycleReq command
0x05	RXParamSetupReq		x	Sets the reception slots parameters
0x05	XParamSetupAns	x		Answers to the RXParamSetupReq command
0x06	DevStatusReq		x	Requests the status of the device
0x06	DevStatusAns	x		Returns the status of the device
0x07	NewChannelReq		x	Creates or modifies the definition of a bidirectional channel
0x07	NewChannelAns	x		Answers to the NewChannelReq command
0x08	RXTimingSetupReq		x	Configures the timing of the reception slots
0x08	RXTimingSetupAns	x		Answers to the RXTimingSetupReq command
0x09	TxParamSetupReq		x	Sets the maximum allowed dwell time and Max Effective Isotropic Radiated Power (EIRD) of the device, based on local regulations

Table 3.5 – List of MAC commands (continuous)

CID	Command Name	Device	Gateway	Description
0x09	TxParamSetupAns	x		Answers to the TxParamSetupReq command
0x0A	DlChannelReq		x	Allows the network to associate a different downlink frequency to the RX1 slot
0x0A	DlChannelAns	x		Answer to the DlChannelReq command
0x0B	RekeyInd	x		Used by an Over-the-Air (OTA) device to confirm security key update
0x0B	RekeyConf		x	Answers to the RekeyInd command
0x0C	ADRParamSetupReq		x	The Network Server uses this to set the ADR_ACK_LIMT and ADR_ACK_DELAY parameters of a device
0x0C	ADRParamSetupAns	x		Answers to the ADRParamSetupReq command
0x0D	DeviceTimeReq	x		A device requests from the network the current date and time
0x0D	DeviceTimeAns		x	Answers to the DeviceTimeReq request
0x0E	ForceRejoinReq		x	The network asks a device to rejoin immediately with optional periodic retries
0x0F	RejoinParamSetupReq		x	The network requests the device to periodically send rejoin messages
0x0F	RejoinParamSetupAns	x		Answers to the RejoinParamSetupReq command
0x80 to 0xFF	Proprietary	x	x	Used for proprietary MAC command extensions

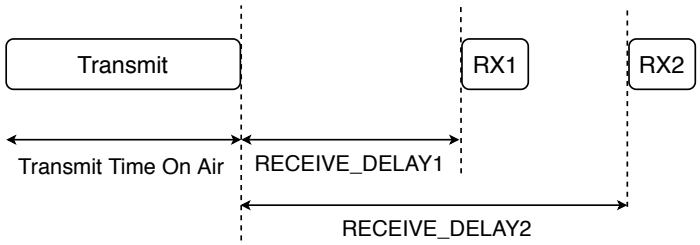

Figure 3.4 – LoRaWAN receive windows.

The first receive window $RX1$ uses a frequency, which is a function of the uplink frequency and a data rate, which is a function of the data rate used for the uplink. $RX1$ opens in $RECEIVE\_DELAY1$ seconds (+/- 20 microseconds) after the end of the uplink transmission. The relationship between uplink and the $RX1$ slot downlink data rate belongs to a specific region and is detailed in the LoRaWAN Regional parameters. By default, the first receive window data rate is identical to the data rate of the last uplink.

The second receive window $RX2$ uses a fixed configurable frequency and data rate, and opens in $RECEIVE\_DELAY2$ seconds (+/- 20 microseconds) after the end of the uplink modulation. The data rate and frequency used can be changed by means of MAC commands. The default data rate and frequency to use belongs to a specific region and is detailed in the LoRaWAN Regional parameters.

The duration of a receive window must be at least the time required by the device radio transceiver to effectively detect a downlink preamble. If a preamble is recognized through one of the receive windows, the radio receiver remains active till the downlink frame is demodulated. If a frame was recognized and then demodulated during the first receive window and the frame was intended for this device after address and MIC checks, the device must not open the second receive window.

If Network Server wants to transmit a downlink message to a device, it needs to initiate the transmission exactly at the beginning of at least one of the two LoRa receive windows. If a downlink message is transmitted during both windows, duplicate frames need to be transmitted in each window.

A device is able to transmit another uplink message if it satisfies one of two conditions: (i) it has received a downlink message in the first or second receiving window of the previous transmission; (ii) the second receive window of the previous transmission has expired.

Table 3.6 – Join Request message fields

Size (bytes)	8	8	2
Join Request	JoinEUI	DevEUI	DevNonce

3.3.6 Device Activation

To join a LoRaWAN network, each device needs to perform an activation. This action can be done in two ways, either by Over The Air Activation (OTAA) or by Activation By Personalization (ABP).

3.3.6.1 Over The Air Activation

In OTAA, devices must follow a join procedure to get involved in data exchanges with a Join Server. Figure 3.5 illustrates the message flow for OTAA. A device initializes this procedure by sending a signed Join Request frame providing the required information for device authentication (i.e., $JoinEUI$, $DevEUI$, $DevNonce$). If the Join Server accepts the device, it derives the session keys and sends a Join Accept with $AppNonce$, $NetID$, $DevAddr$ along with other information such as downlink parameters, $RxDelay$ (i.e., the delay for waking up for receiving an ACK), and a list of channels to use. Based on $AppNonce$, $NetID$, $DevAddr$, the device derives the session keys and can start sending application traffic. The join procedure consists of either a Join Request or ReJoin Request and a Join Accept exchange.

Join Request Message holds $JoinEUI$ (i.e., an ID in the IEEE EUI64 address space used to identify uniquely the Join Server), $DevEUI$ (i.e., a global device ID in the IEEE EUI64 address space used to identify uniquely the device), and $DevNonce$ as in Table 3.6. $DevNonce$ is a counter beginning at 0 when the device is at first powered up and then increased with every Join Request. The Join Server observes the last $DevNonce$ value used by each device and ignores Join Request if $DevNonce$ is not increased. The Join Request message is not encrypted and can be sent using any data rate and under a random frequency hopping sequence across the specified join channels.

Join Accept Message contains a server nonce ($AppNonce$), a network identifier ($NetID$), a device address ($DevAddr$), some downlink parameters ($DLSettings$), and

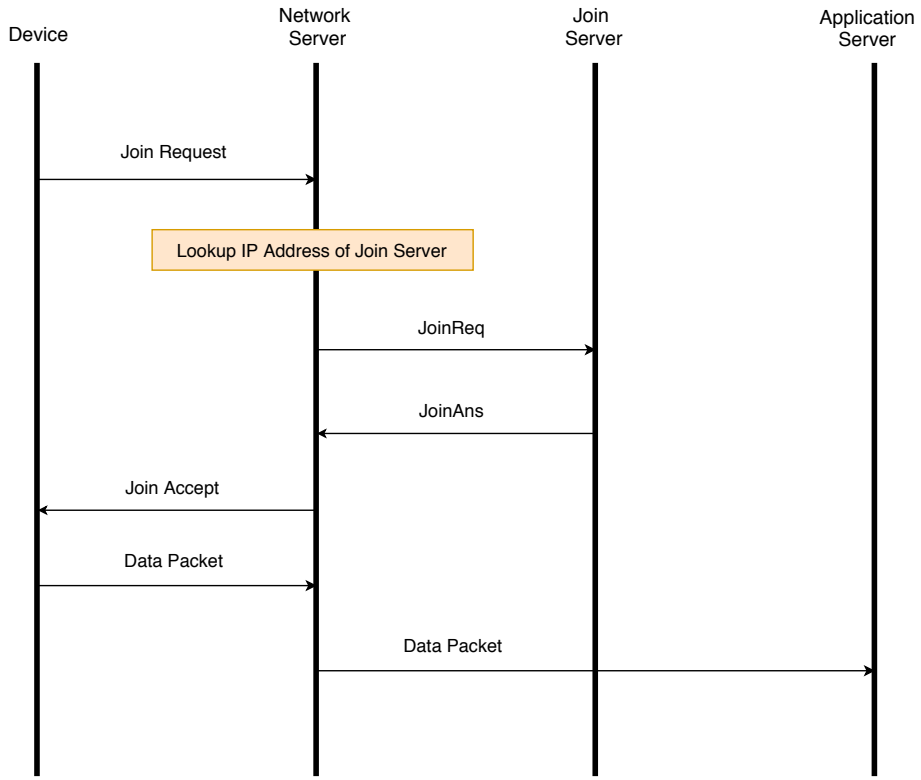

Figure 3.5 – LoRaWAN OTAA procedure.

Table 3.7 – Join Accept message fields

Size (bytes)	3	3	4	1	1	16 (Optional)
ReJoin Accept	AppNonce	Home_NetID	DevAddr	DLSettings	RxDelay	CFList

an optional list of network parameters ($CFList$) of the network that the device is joining (see Table 3.7). This message is sent by the Join Server to respond to the Join Request message or ReJoin Request message if the device is allowed to join a network. The $AppNonce$ is a specific counter value given by the Join Server and used by the device to derive the session keys. Like $DevNonce$ field on Join Request, $AppNonce$ is increased with every Join Accept message.

ReJoin Request Message is sent by the device to initialize the rejoin procedure. Once activated, a device can periodically send a ReJoin Request to the Join Server. This message allows the backend the chance to initialize a new session context for the device. According to three different purposes, there are three types of ReJoin Request messages that a device can be sent. ReJoin Request message type 1 contains $JoinEUI$

Table 3.8 – ReJoin Request Type 1 Message Fields

Size (bytes)	1	8	8	2
ReJoin Request	ReJoin Type = 1	AppEUI	DevEUI	RJcount1

Table 3.9 – ReJoin Request Type 0 or 2 Message Fields

Size (bytes)	1	3	8	2
ReJoin Request	ReJoin Type = 0 or 2	NetID	DevEUI	RJcount0

and *DevEUI* (see Table 3.8). Similarly to the Join Request message, but this message may be transmitted on top of normal applicative traffic without disconnecting the device.

Different from ReJoin Request 1 message type, ReJoin Request message type 0 or 2 contains *NetID* and *DevEUI* (see Table 3.9). While the ReJoin Request message type 0 is used to reset a device context including all radio parameters, the ReJoin Request message type 2 used to rekey a device or change its *DevAddr*. The ReJoin Request message is not encrypted.

3.3.6.2 Activation By Personalization

In ABP, *DevAddr* and session keys are stored directly to the device instead of being derived during the join procedure. The device in this context is provided with all the required information for joining in a specific LoRa network once it is started.

When an ABP device accesses the network for the first time or after a reinitialization, it will transmit the *ResetInd* MAC command in the *FOpt* field of all uplink messages until it receives a *ResetConf* command from the network.

Figure 3.6 shows the activation of an ABP device with a Network Server. This procedure applies to both devices and networks. First, the device, Network Server, and Application Server are configured with the required information so that the device can send packets as soon as it is powered on.

Then, when the device has an application payload to send, it can do so without performing any setup signalling with the network. The packet includes the application

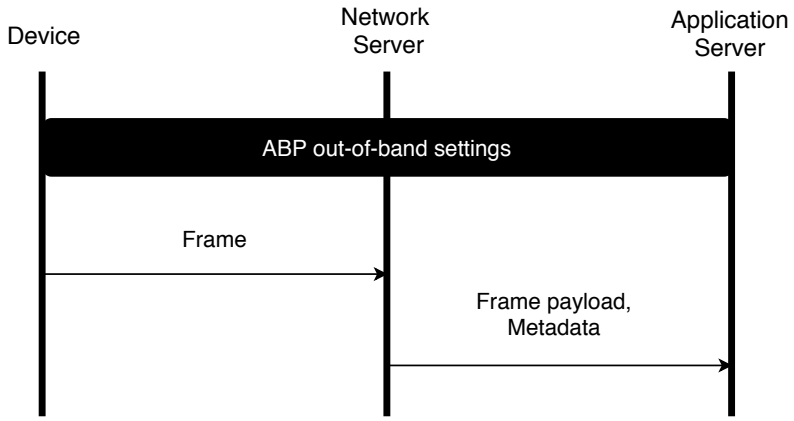

Figure 3.6 – ABP activation procedure.

payload encrypted using $AppSKey$ and with MIC generated using the network session integrity keys. When the Network Server receives the packet, it will perform network session integrity key lookup based on $DevAddr$ of the received packet. The Network Server will verify MIC using the retrieved keys. If the keys are not found, or if MIC verification fails, the Network Server will drop the packet.

Finally, the Network Server will send the encrypted payload of the accepted packet to the Application Server associated with the device. The application payload may be accompanied by the metadata, such as $DevAddr$, $FPort$, $timestamp$, etc. The Network Server will consider receipt of the very first packet from the device as the activation of a LoRa session for the device.

3.3.7 Deactivation of OTAA devices

The LoRa session of an OTAA device can also be terminated for various reasons, such as the user reaching the end of contract, malicious device behaviour, etc. The procedure used for deactivating the session is called the Exit procedure, which is the counter-part of the join procedure.

There is no explicit and dedicated LoRaWAN signalling for performing the Exit procedure. It is assumed that the device and the backend rely on application-layer signalling to perform this procedure.

The device successfully performing a new Join/Rejoin procedure also terminates the current LoRaWAN session, and in a way, it can be considered as the deactivation

associated with that session.

3.4 LoRa Network Reference Model

A typical LoRa network can be viewed as a "star-of-stars" architecture. Figure 3.7 shows the Network Reference Model for this LPWA network [41].

- **LoRa Device (End Device, End Point, End Node)**. The LoRa device is a sensor or an actuator. The device communicates with a LoRaWAN network through Gateways. The application layer of the device communicates with a specific Application Server. All application layer payloads of this device are routed to its corresponding Application Server.

- **Gateway**. Unlike cellular a network in which mobile devices are associated with the serving base stations, data packets transmitted by LoRa devices are sent to all gateways that relay packets between devices and the Network Server. Devices send packets to gateways over a single wireless hop and gateways connect to the Network Server over an IP back-bone.

- **Network Server**. The Network Server is the centre of the star topology. It is responsable for:

  - LoRa device address check,
  - Frame authentication and frame counter checks,
  - Sending acknowledgements,
  - Data rate adaptation,
  - Answering all MAC requests coming from LoRa devices,
  - Forwarding uplink application payloads to the specific Application Servers,
  - Forming a queue of downlink payloads coming from any Application Server to any device connected to the LoRa network,
  - Forwarding Join Request and Join Accept messages between devices and Join Servers.

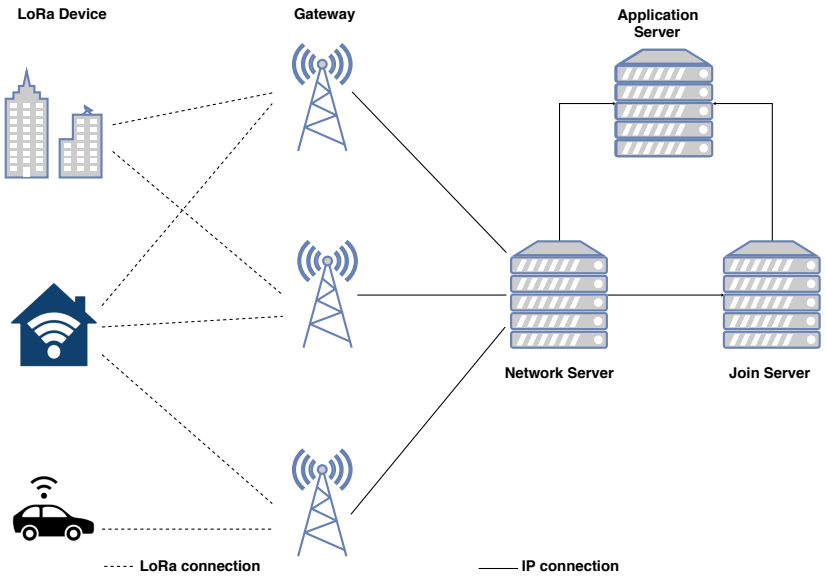

Figure 3.7 – LoRaWAN network architecture.

- **Join Server**. The Join Server manages the OTAA activation process of the LoRa device. There may be several Join Servers connected to a Network Server and a Join Server may connect to several Network Servers.

  A LoRa device signals which Join Server should be interrogated through the $JoinEUI$ field of the Join request message. Each Join Server is identified by a unique $JoinEUI$ value. The Join Server contains the required information to process uplink Join request frames and generate the downlink Join accept frames. It also performs the network and application session key derivations. It communicates the Network session key of the LoRa device to the Network Server, and the application session key to the corresponding Application Server. For that purpose, the Join Server will contain the following information for each LoRa device under its control:

  - $DevEUI$
  - $AppKey$
  - $NwkKey$ (only applicable to the LoRa Device on LoRaWAN 1.1)
  - Application Server identifier
  - A way to select the preferred network in case several networks can serve the LoRa device.
  - LoRaWAN version of the LoRa device (i.e., LoRaWAN 1.0, 1.0.2, or 1.1)

The Join Server can be able to establish secure communication with Network Server or Application Server to provides end-point authentication, confidentiality, integrity and replay protection. The Join Serve can also be able to securely deliver Application Session Key to the Application Server. The Join Server may be connected to several Application Servers, and an Application Server maybe connected to several Join Server.

- **Application Server**. The Application Server handles all the application layer payloads of the associated LoRa devices and provides the application-level service to the end user. It also generates all the application layer downlink payloads towards the connected LoRa devices.

3.5 Adaptive Data Rate

To maximize both battery lifetime of the devices and overall network capacity, the LoRa network can control the data rate for each device with the Adaptive Data Rate (ADR) scheme [1]. When the ADR feature is enabled, the network will use the fastest data rate possible.

The ADR algorithm is executed on the Network Server and is managed by the LoRa network operator. This algorithm controls the data rate used by each device for its uplink and downlink based on a set of related LoRaWAN MAC commands (i.e., *LinkADRReq* and *LinkADRAns*; *ADRParamSetupReq* and *ADRParamSetupAns*). With the *LinkADRReq* command, the Network Server requests a device to perform a data rate adaptation. The device then answers to the *LinkADRReq* with a *LinkADRAns* commands. With the *ADRParamSetupReq* command, the Network Server changes the ADR_ACK_LIMIT and the ADR_ACK_DELAY parameters of a device. The device then use the *ADRParamSetupAns* command to acknowledge the reception of the *ADRParamSetupReq* command.

There can be different ADR algorithms running concurrently for different groups of devices that have different application profiles. For example, a water meter and a smart light controller might not be controlled by the network in the same style. As all complex and intelligence stays in the Network Server, the device only executes the MAC commands received from the Network Server. Because of this, the ADR algorithm can

be updated easily as the Network Server when needed. There is nothing to reconfigure on the device.

3.6 Collisions in LoRa

Collisions occur when two devices transmit packets at the same time. However, a receiver can correctly receive a frame in the presence of interfering signals, because the LoRa physical layer is robust enough to resist significant interference. Haxhibeqiri et al. [3] used a simulation model based on the measurements of the interference behavior between two motes with a duty cycle of 1% to show that when their number increases to 1000 per Gateway, the packet loss rate increases to 32%. However, this level of the loss rate should be considered as low compared to 90% in pure ALOHA for the same load and it results from taking into account the capture effect.

Two transmissions overlapping in time are *quasi-orthogonal*: when device i sends a packet using SF_i and the transmission of device j with SF_j overlaps, the reception of the packet from device i is successful if SIR (Signal to Interference Ratio) of the transmitted signal i is above a_{ij} [dB], the SIR threshold required for rejecting the interfering signal, i.e., the margin a packet sent at SF_i must have for successful reception (BER of 1%) if the interference packet is sent at SF_j (see Table 3.10 [42]).

Table 3.10 – SIR interference rejection thresholds in [dB].Transmitter: SF_i, interferer: SF_j.

SF_i \ SF_j	7	8	9	10	11	12
7	1	-8	-9	-9	-9	-9
8	-11	1	-11	-12	-13	-13
9	-15	-13	1	-13	-14	-15
10	-19	-18	-17	1	-17	-18
11	-22	-22	-21	-20	1	-20
12	-25	-25	-25	-24	-23	1

3.7 LoRa Performance

3.7.1 Analysis of LoRa capacity and limitations

Several authors studied the issue of limits to the capacity of LoRa and its scalability to a large number of devices. Augustin et al. [6] presented throughput measurements on a testbed showing: i) less than 10% of loss rate over a distance of 2 km for SF9-SF12 and ii) more than 60% of loss rate over 3.4 km for SF12. They also simulated the LoRa behavior for a larger number of devices and showed that it behaves closely to ALOHA with the maximum channel capacity of 18% and an increasing collision ratio: for a link load of 0.48, the ratio is around 60%.

Adelantado et al. [7] explored LoRa from the point of view of the capacity and the network size. They observed that for low duty cycles, throughput is limited by collisions, whereas for higher duty cycle values, the maximum duty cycle set by the ETSI regulations prevents devices from increasing their packet transmission rates and limits throughput. For instance, for 1000 devices, the maximum packet rate per node is 38 pkt/hour (packets of 50 B) with the probability of successful reception of only 13%.

Reynders et al. [8] compared the performance of LoRa and Ultra Narrowband (UNB, SIGFOX-like) networks with regard to the range and coexistence. They showed that UNB MAC is slightly better than LoRaWAN: the latter discards both colliding packets at reception, while the UNB network enables reception of the strongest packet thanks to the capture effect. The maximal throughput of the network occurs for 10^5 devices

in the network, but results in a packet loss of 63%.

Haxhibeqiri et al. [3] investigated the scalability of LoRa in terms of the number of devices per gateway. They used a simulation model based on the measurements of the interference behavior between two nodes to show that when the number of nodes with the duty cycle of 1% increases to 1000 per gateway, losses increase to 32%. However, this level of the loss rate should be considered as low compared to 90% in pure ALOHA for the same load and it results from taking into account the capture effect, which apparently plays an important role in the LoRa behavior.

Mikhaylov et al. [9] showed that a LoRa cell can potentially serve a large number of devices, but devices are limited to sending only a few bytes of data per day. The majority of devices need to be located in the vicinity of the gateway: only less than 10% can reside at distances longer than 5 km. Another factor that limits scalability is the use of acknowledgements—as the gateway is subject to the same ETSI restrictions on the duty cycle, it cannot acknowledge each packet in a dense network.

Bor et al. [10] developed a LoRa simulation to study its scalability. They showed that a typical Smart City deployment can support 120 nodes per 3.8 ha, which is not sufficient for future IoT deployments. Other studies in the literature analyzed the performance of the LoRa modulation—Goursaud and Gorce [11] considered other technologies (SigFox, Weightless, and RPMA by Ingenu) in addition to LoRa to highlight their pros and cons.

Ochoa et al. [12] proposed various strategies to adapt LoRa radio parameters to different deployment scenarios. Their simulation results showed that in a star topology, we can achieve the optimal scaling-up/down strategy of LoRa radio parameters to obtain either a high data rate or a long range while respecting low energy consumption.

Lone et al. [43] designed WiSH-WalT, a framework for controllable and reproducible LoRa testbeds. They have set up a series of experiments with a LoRa device sending packets to several public TTN gateways around. They showed that a good level of PDR ($\geq$ 80%) is only achieved for these gateways with the distance from devices is less than 1.2 km almost independently of SF and TP. For other gateways, the impact of TP is as expected as PDR increases with TP. In addition, they observed that a very good level of signal-to-noise ratio (SNR) turns into a very good level of PDR. For some gateways, however, good levels of SNR only lead to an average PDR.

Georgiou and Raza [5] provided a stochastic geometry framework for modeling the performance of a LoRa network with single gateway. They indicated that the coverage probability drops exponentially when the number of contending devices increases. They concluded that LoRa networks will become interference-limited rather than noise-limited in dense deployment scenarios because of the LoRaWAN access method.

Duda and Heusse [44] proposed to analyze the LoRa performance under an assumption of inhomogeneous density of nodes: it decreases with the inverse square of the distance to the gateway. They have used the model by Georgiou and Raza [5] to analyze the capacity of a LoRaWAN cell for different types of SF allocations: equidistant, PDR-based, and SNR-based. Based on the numerical results, they concluded that:

- For a target range and a required PDR level, it is possible to find an allocation of annuli l_j that leads to the maximal number of nodes that benefit from the PDR level.
- There is a trend towards configurations made up smaller cells that focus on nodes close to the gateway. In this way, nodes benefit from low SF.
- To give to more nodes with the required PDR level, it needs to take into account the context of multiple gateways that will improve the overall capacity while maintaining low energy consumption.

Attia et al. [45] presented the results of extensive experiments in The Things Network (TTN) to analyze the quality of LoRa links by calculating Packet Reception Rate (PRR) based on the payload length. The results indicated that there is only a small impact of the payload length on PRR. It means that the bit error rate as a result of the ambient noise at the receiver and collisions are not the only factors that influence the packet reception probability. Besides that, their measurements proved that the behaviour of LoRa channel is similar to a slow fading Rayleigh channel. In addition, they concluded that P_s depends on SNR and SF, and often becomes a dominant factor of reception successfully depending on the signal strength at a gateway.

Caillouet et al. [46] proposed a theoretical framework for maximizing the LoRaWAN capacity in terms of the number of nodes. The model allocates the spreading factor to the nodes optimally so that attenuation and collisions are optimized. They used a

propagation model considering the Rayleigh channel and took into account the physical capture and imperfect SF orthogonality while guaranteeing a given transmission success probability to each served node in the network. The numerical results showed the effectiveness of their SF allocation policy. Their framework also quantifies the maximum capacity of single cell networks and the gain induced by multiplying the gateways on the covered area.

Heusse et al. [47] proposed a model for the capacity of a LoRaWAN cell that takes into account collisions and the capture effect. It allows assessing the number of nodes that can share a single LoRaWAN cell for a given node density, cell range, traffic generation rate, and target PDR. They developed an expression for PDR based on the assumption that overcoming a collision and the ambient noise have dependent probabilities. In addition, their hypothesis of similar traffic intensity for all nodes puts under the spotlight the problem of suitable SF allocation. This problem is critical for dense deployments in a short range cell, in which it greatly helps to distribute as much traffic as possible on lower SFs.

3.7.2 Analysis of proposals for improving LoRa performance

In this part, we review some proposals for improving LoRa performance, especially with the approaches related to our Timemaps proposed scheme.

Several authors considered the issue of scheduling transmissions in LoRa. Rizzi et al. [48] introduced a TSCH-like scheduling scheme for LoRa. The authors set up several experiments on a testbed consisting of three devices, a Gateway, and a Network Server. The network achieves the theoretical maximum LoRaWAN capacity when the synchronization scheme is applied on all channels and all SFs are used by the devices. However, the authors do not present the mechanisms for slot and SFs allocation. Moreover, they do not take into account energy consumption.

Reynders et al. [49] presented a MAC layer based on a two-step scheduling scheme that allows a device determining its transmission power, SF, and when and on which channel to transmit based on a schedule provided by the Gateway. However, they showed that the reliability improvement compared to LoRaWAN is not significant—

about 5% fewer losses for 3500 devices in a scenario with multiple Gateways.

Piyare et al. [50] proposed an on-demand Time Division Multiple Access (TDMA) protocol for IoT for improving both energy consumption and latency. Based on an indoor testbed composed of 11 sensor nodes, they achieved PDR of 100% by eliminating the possibility of packet collisions. The authors, however, did not test the proposed scheme for a network with a large number of nodes and over long distances. They did not provide a mechanism for effective use of the Gateway that simultaneously supports multiple channels and multiple SFs.

Haxhibeqiri et al. [51] defined a synchronization and scheduling scheme for LoRaWAN networks. At the Network Server, the Network Synchronization and Scheduling Entity (NSSE) schedules uplink and downlink traffic for end nodes by means of a central scheduler. Each device, before being able to transmit data packets, has to request time slots by contacting NSSE that sends the time slot indices encoded in Bloom filters. The authors show that PDR increases by 7% and 30% for SF7 and SF12, respectively. However, instead of scheduling transmissions for all SFs, the algorithm only considers individual SF (i.e., it takes a single-SF as an input). It is limited by the fact that all devices must use the same SF.

Zorbas et al. [52] proposed two offline algorithms for allocation of SFs and slots to nodes in case of bulk data transmissions. They define a frame as a 2D array with six rows, one per SF, and each consisting of a list of slots. A scheduling algorithm finds a frame with the best allocation per node or transmission to minimize the data collection time. The Global algorithm computes the schedule for all transmissions and has a single frame. The Light algorithm provides a periodic schedule consisting of repeated frames. The algorithm operation assumes that "the nodes are periodically synchronised by the gateway according to a global clock", but the authors do not explain how it is done. The validation through simulation assumes 500 kHz bandwidth, no capture effect (i.e., two or more transmissions collide when they overlap in time), and shows 100% PDR for up to 1000 nodes along with much lower energy consumption compared with LoRaWAN, which is surprising because even if scheduling lowers collision ratio, it increases energy consumption with an important overhead for transmitting schedules.

3.8 Conclusion

In this chapter, we have presented the details of the LoRa physical layer: the parameters of the modulation, the data rate, frame size limitations, spreading factors, and receiver sensitivity. Concerning LoRaWAN, we have introduced different classes of devices, the formats of MAC messages, and the list of MAC commands. We have presented the Network Reference Model for LoRaWAN architecture and explained the process of device activation in LoRa networks. Finally, we have discussed various analyses of LoRa performance and presented proposals for improvement with transmission scheduling.

Chapter 4

NS-3 LoRa Module

Contents

4.1 Introduction

In this chapter, we first introduce the NS-3 network simulator software. We then show the details of the development of the LoRa module in NS-3.

4.2 NS-3 Overview

NS-3 is an open source discrete-event network simulator under the GNU GPLv2 license, publicly available for networking research and education [53]. It provides models of different types of networks and supports users in performing simulation experiments. Figure 4.1 shows the NS-3 network architecture. The key concepts of modeling networks in NS-3 are presented below:

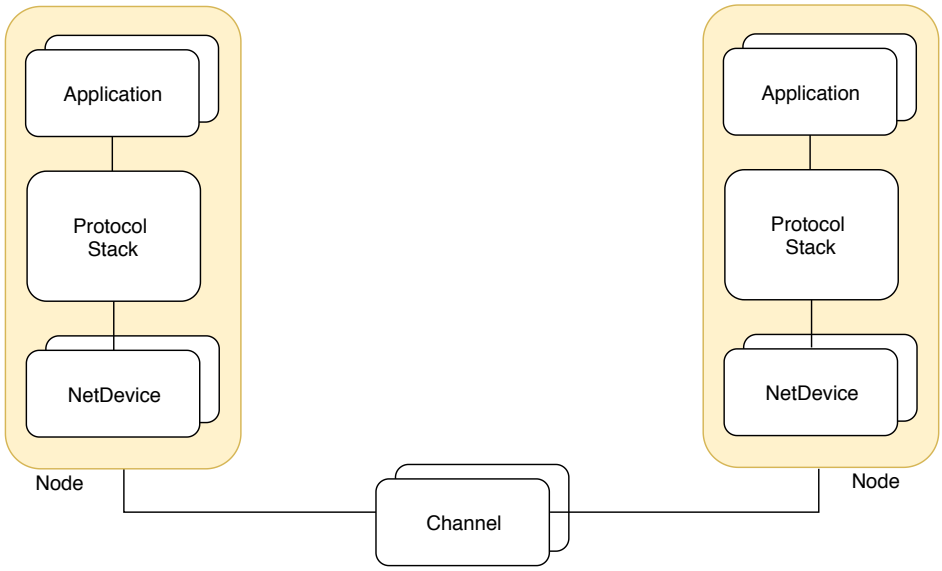

Figure 4.1 – NS-3 network architecture.

- **Node:** in NS-3, a node is a computing device similar to a computer. It is represented in C++ by the Node class. The Node class gives methods for managing the instants of computing devices during simulations.

- **Application:** it embodies all software above network protocols—there is neither real concept of an operating system, nor concept of privilege levels or system calls in NS-3. It runs on the NS-3 Node to perform tasks in the simulation. This object is represented in C++ by the Application class. The Application class provides methods for managing the instants of user-level applications in simulations.

- **Channel:** provides methods for managing communication sub-network objects. Each communication sub-network object is called the channel and is represented in C++ by the Channel class. Nodes then connect to channels to establish a network in simulations. In more detail, the channel is a physical connector between a set of NetDevice objects.

- **NetDevice:** like a Network Interface Card in real computers, the NetDevice in NS-3 already integrates a software driver. It means that the NetDevice will work without installation of a software driver to control the simulated hardware. In the world of NS-3 simulation, a NetDevice is installed in a Node object to enable the Node to link with other Nodes over Channels.

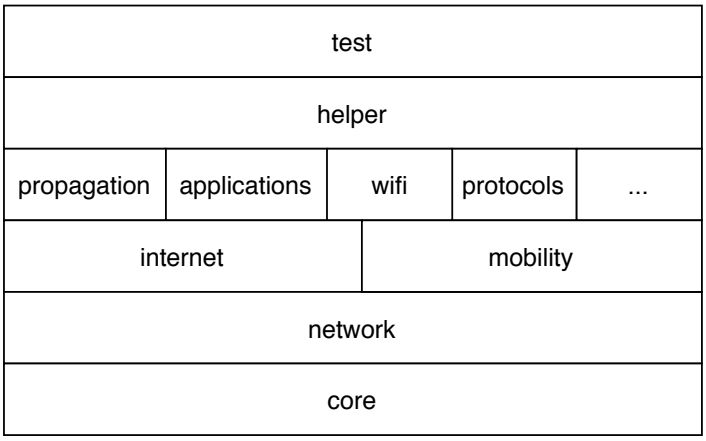

Figure 4.2 – LoRa NS-3 software organization.

4.2.1 NS-3 Software Organization

In NS-3, the source code is arranged in the *src* directory. The structure of source code is shown in the diagram in Figure 4.2. The core of the simulation contains of the components commonly used by all protocol, hardware, and environmental models. The core module is achieved in *src/core*. The network module represents packets in the simulation and is achieved in *src/network*. These two important modules become a generic core that can be used by different kinds of networks, not just Internet-based networks. The above modules of NS-3 such as *internet*, *mobility*, *propagation*, *applications*, $wifi$, *protocols*, etc. are independent of specific network and device models.

In addition to the above core, NS-3 also provides two modules to support users and developers easier in building simulation scripts and adding new modules. First, the *helper* API makes us more comfortable to build NS-3 script with the same power of the low-level interface. Second, the *test* framework provides tools for automating the process of validation and verification the code in test suites to help quickly identify possible regressions.

4.2.2 NS-3 Logging

The NS-3 logging system is used to track or debug simulation programs. Logging output can be controlled in two ways. The first one is setting the NS_LOG environment variable, as below:

```
$ NS_LOG="*" ./waf --run my-first-example
```

This command will run the *my-first-example* program with all logging output.

The second one for enabling logging is to use statements in the *main*() function, such as in a *csma* example as below:

```
int
main (int argc, char *argv[])
{
     LogComponentEnable ("CSMAExample", LOG_LEVEL_INFO);
     ...
}
```

The $CSMAExample$ here is a log component. To use it, we first need to register this component in the logging system by using $NS\_LOG\_COMPONENT\_DEFINE(...)$ command as below:

```
using namespace ns3;
NS_LOG_COMPONENT_DEFINE ("CSMAExample");
...
```

The LOG_LEVEL_INFO is a level in the logging system and belongs to a severity class. The severity classes are defined as enum constants in Table 4.1. In addition, the logging levels are defined inclusively to display log messages at a given severity class and higher, as presented in Table 4.2.

4.2.3 NS-3 Tracing

There are several methods to bring out needed messages of an NS-3 program. The most simple one is to print the information by the command *std* :: *cout*, as below:

Table 4.1 – NS-3 logging severity classes

Severity Class	**Meaning**
LOG_NONE	The default, no logging
LOG_ERROR	Serious error messages only
LOG_WARN	Warning messages
LOG_DEBUG	For use in debugging
LOG_INFO	Informational messages
LOG_FUNCTION	Function tracing
LOG_LOGIC	Control flow tracing within functions

Table 4.2 – NS-3 logging levels

Level	**Meaning**
LOG_LEVEL_ERROR	Only LOG_ERROR severity class messages
LOG_LEVEL_WARN	LOG_WARN and above
LOG_LEVEL_DEBUG	LOG_DEBUG and above
LOG_LEVEL_INFO	LOG_INFO and above
LOG_LEVEL_FUNCTION	LOG_FUNCTION and above
LOG_LEVEL_LOGIC	LOG_LOGIC and above
LOG_LEVEL_ALL	All severity classes

```
#include <iostream>
using namespace std;
...
int main ()
{
    uint32_t a = 5;
    ...
    std::cout << "The value of a is :" << a << std::endl;
    ...
    return 0;
 }
```

However, the command *std::cout* is suitable for small environments. When our simulations become more complex, NS-3 provides a mechanism for logging with several control over output via the NS_LOG environment variable, but the level of control is not very fine grained at all. In addition, as NS_LOG output is only available in debug builds, we are not able to obtain log output from the best builds, which run about twice as fast.

For these reasons, the tracing system in NS-3 becomes one of the most important mechanisms that allows a simulation to generate output data for further study. This system is based on the independence of tracing sources and tracing sinks, together with a mechanism for connecting between them.

Trace sources are used to notice events that happen in a specific simulation and give access to fundamental data. For example, a trace source can specify at what time one packet is received by a net device and give access to the packet contents for trace sinks. Trace sources play a role of event generators and trace sinks are the one that consume trace information.

NS-3 provides trace helpers that wrap the low-level tracing system to help us in configuring and selecting different trace events and writing them to files. There are two type of trace helpers in NS-3, namely PCAP Tracing Device Helper and ASCII Tracing Device Helper.

PCAP (Packet CAPture) Tracing Device Helper is an API that includes the defini-

tion of a *.pcap* file format. There are many traffic trace analyzers that use this packet format, like Wireshark. The command used to active PCAP tracing in a simulation script is as below:

```
...
csma.EnablePcapAll ("mycsma");
...
```

Then, when we run our simulation script, trace files with the prefix *mycsma* are generated at the top level directory of NS-3.

With ASCII Tracing Device Helper, the simulation script can generate trace files in ASCII format. By including the following code line in our simulation script, we enable ASCII tracing on all CSMA devices in the script:

```
...
csma.EnableAsciiAll (ascii.CreateFileStream ("mycsma.tr"));
...
```

With the *mycsma.tr* trace file that generating by running the script, we can see its content by our favorite editors.

4.2.4 NS-3 Testing Framework

To support verification and validation, NS-3 provides a testing framework for developers, namely *buildbots* (i.e., build robots). The *buildbots* is an automated system that allows us to rebuild and test NS-3 every day. The *buildbots* uses a Python program, called *test.py* to run all of the tests and then collect the test reports to developers.

The *test.py* program is very flexible in allowing the developers to set the kind and number of tests to run, the kind and amount data to generate. For example, when we execute the following test, *test.py* will do all accessible test cases and build test report in a compact form:

```
$ ./test.py
```

This test program will print out test results that comprise names of test suites, followed by number of *PASSED*, *SKIPPED*, *FAILED*, *CRASHED* or *VALGRIND ERRORS* indications:

```
Waf: Entering directory ‘/home/drakkar/ns-allinone-3.27/ns-3.27/build’
[ 959/1481] Compiling src/lora/examples/lora-example.cc
[1466/1481] Linking build/src/lora/examples/ns3.27-lora-example-debug
Waf: Leaving directory ‘/home/drakkar/ns-allinone-3.27/ns-3.27/build’
Build commands will be stored in build/compile_commands.json
’build’ finished successfully (2.711s)

PASS: TestSuite callback
PASS: TestSuite build-profile
PASS: TestSuite attributes
...
PASS: TestSuite steady-state-rwp-mobility-model
PASS: TestSuite threaded-simulator
PASS: TestSuite packet
111 of 111 tests passed
  (111 passed, 0 skipped, 0 failed, 0 crashed, 0 valgrind errors)
```

4.3 LoRa Simulation Module in NS-3

We have developed an NS-3 module that simulates the LoRa behavior. Figure 4.3 shows its architecture. In this section, we mainly focus on four important parts of the LoRa module, namely, LoRa Device, LoRa Gateway, LoRa Channel, and Network Server.

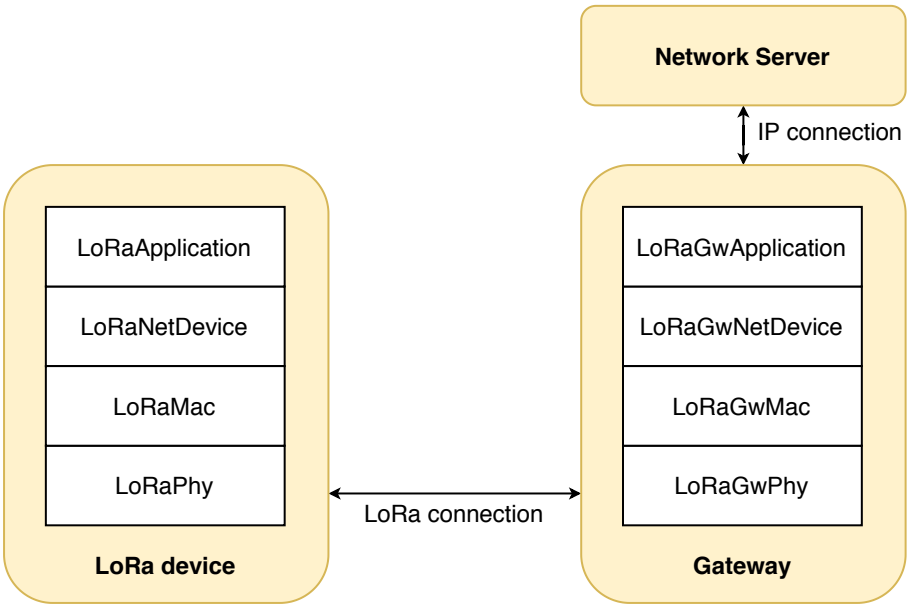

Figure 4.3 – LoRa NS-3 module architecture.

4.3.1 LoRa Device

We have developed the LoRa Device of our module based on the following parts and documentation:

- Knight implemented an open source physical layer of LoRa for end devices [54]. As this source code is written in C and Python, and provided with documentation, it is a very good reference for implementing the physical layer of LoRa module in NS-3.
- Blum et al. [55] also implemented the physical layer of LoRa by making use of software-defined radio hardware to receive and decode LoRa. It supports the data format used by the SX1272 LoRa chip.
- Semtech LoRa device Datasheet provided by Semtech Corporation [56].
- LoRaWAN Specification v1.1 [1].

The **LoRaPhy** class models the LoRa physical layer—it simulates the behavior of the SX1272 LoRa chips in LoRa devices. The class uses a variable nameed *m_state* representing all states of the transceiver that can take one of the following values:

- **TX**: when the device transmits a packet.

- **RX**: when the device receives an incoming packet.
- **IDLE**: when the device is available for transmission of its packets or receiving packets from other objects in the network.
- **SLEEP**: when the device goes to a sleep mode to save energy.

When the device has to send a packet, the LoRaPhy class takes the packet from LoRaWAN MAC layer of the device and delivers it to the LoRa Channel class.

The **LoRaMac** class models LoRaWAN MAC layer of a *class A* LoRaWAN device. This class defines an access method similar to ALOHA: a device wakes up and sends a packet at once. We used LoRaWAN Specification v1.1 to build this class.

4.3.2 LoRa Gateway

We have developed the LoRa Gateway of our module based on the following parts and documentations:

- The library corresponding to the driver/hardware abstraction layer for building a gateway using a concentrator board based on the Semtech SX1301 multi-channel modem [57]
- LoRaWAN Specification v1.1 [1].

The **LoRaPhyGw** class models the LoRa physical layer of a LoRa Gateway. It inherits from the LoRaPhy class. LoRaPhyGw is the one that decides if a packet obtained from the channel is correctly received based on its signal power and the interference from others. This class takes the packet from the LoRa Channel class and delivers it to the LoRaWAN MAC layer of the Gateway.

We need the the **LoRaMacGw** because of the feature of the forward-only MAC layer. In addition, this class is used for working with MAC commands that are either piggybacked in the FOpts field (for the small size of sequence) or contained in the FRMPayload (for the big size of sequence).

We implement the capture effect in the LoRa Gateway that respects three following conditions:

- If a packet interferes with the transmission with the same SF, then the packet is captured if it is received with 6 dB more power than the colliding frame [58].
- If a packet interferes with the transmission with different SF, then the packet is captured if it is received with SIR above the SIR threshold required for rejecting the interfering signal (see Table 3.10 [42]).
- If more than two packets overlap the transmitted one, we consider it as lost.

4.3.3 LoRa Channel

For the LoRa Channel, we use the Spectrum Channel providing support for modeling the frequency-dependent aspects of communications in NS-3. The module provides:

- a list of classes for simulating signals,
- an interface between spectrum channel and spectrum PHY namely Channel/PHY interface. It is based on a signal representation of power spectral density that is independent with technology,
- two implementations of the channel that technology-independent based on the Channel/PHY interface,
- basic implementations of PHY based on the Channel/PHY interface.

The spectrum Channel/PHY interface is defined by the base classes: *SpectrumChannel* and *SpectrumPhy*. The interaction between them simulates the transmission and reception of signals over the channel.

The Spectrum module provides two *SpectrumChannel* implementations: *SingleModelSpectrumChannel* and *MultiModelSpectrumChannel*. They both provide the following functionality:

- Modeling of the propagation loss in two forms: (i) we can add models based on $PropagationLossModel$ on these channels. Only linear models (in which the loss value does not depend on the transmission power) can be used. These models are single-frequency in the sense that the loss value is affected equivalently

to all elements of the power spectral density. (ii) we can add models based on *SpectrumPropagationLossModel* on these channels. These models can be frequency-dependent in the sense that the separate loss value is counted and affected to each element of the power spectral density.

- Modeling the propagation delay by adding a model based on *PropagationDelayModel*. The propagation delay is frequency-independent applied to the signal as a whole. Delay modeling is implemented by scheduling the *StartRx* event (i.e., Start Reception event) with a delay respect to the *StartTx* event (i.e., Start Transmission event).

4.3.4 Network Server

We have developed the Network Server based on LoRaWAN Backend Interfaces 1.0 Specification [41]. This module also includes the role of a Join Server to manage the OTAA activation process of the LoRa device.

4.4 Energy Framework in NS-3

We use the energy framework implemented in NS-3 by Wu et al. [59] to estimate energy consumption at a battery powered node or in the whole network. Figure 4.4 shows the NS-3 energy framework structure including energy sources, device energy models, and the interfaces interconnecting them. Energy source models represent different types of energy sources, such as Lithium-ion batteries and others. Device energy models represent components of a node, for example, a WiFi radio, which consumes energy from the energy source.

The energy source is the power supply or batteries of network nodes. A network node is able to have more than one energy sources, and each energy source can connect to more than one device energy models. When we connect an energy source to a device energy model on a node, it means that the node draws power from the source. The main functionality of the energy source is to supply energy for devices on the node. When energy is totally drained from the energy source, it alerts the devices on the node so that each device can react to this event. Moreover, each node can access objects of

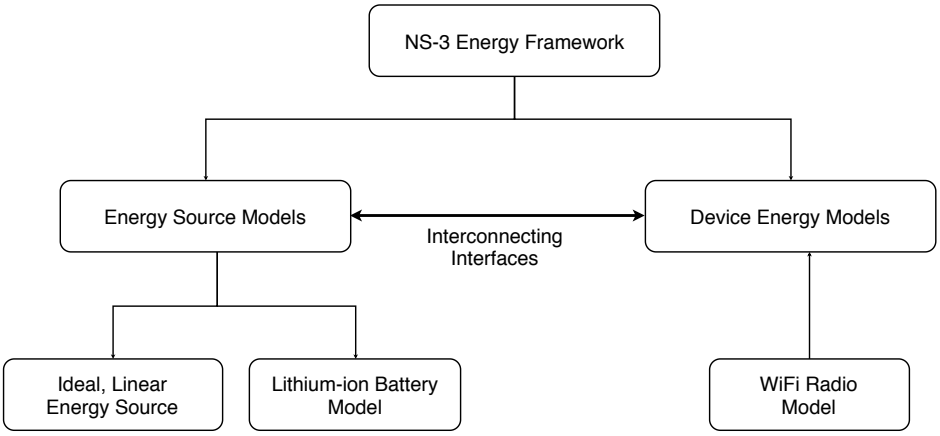

Figure 4.4 – NS-3 Energy Framework Structure.

the energy source to get more information such as remaining energy, energy fraction (i.e., battery level), etc. In NS-3, the energy sources supply power to devices on each node at a constant voltage of 3.3 V.

Our LoRa module uses the energy consumption model in which all operations of devices are represented as states with their associated current draw values that determine power consumption. In case of the radio, we assume three states defined as *transmit*, *receive*, and *sleep*. The total energy consumption E is composed of the energy consumed in each state denoted as E_{tx}, E_{rx}, and E_s, respectively:

$$E = E_{tx} + E_{rx} + E_s, \tag{4.1}$$

$$E = T_{tx}P_{tx} + T_{rx}P_{rx} + T_sP_s, \tag{4.2}$$

where T_{tx}, T_{rx}, T_s, and P_{tx}, P_{rx}, P_s are time spent and power consumption in the states for transmission, reception, and sleep, respectively. The energy consumption model reflects the operation of duty cycle MAC layers in a realistic way [60, 61].

4.5 Adding a new module to NS-3

In NS-3, all modules are organized in the *src* folder. Each module is arranged in a folder that has the same name of the module [62]. For example, the *csma* module can be found in the *src/csma* folder. The directory structure and required files of this module are as follows:

```
src/
```

```
csma/
     bindings/
     doc/
     examples/
              wscript
     helper/
     model/
     test/
             examples-to-run.py
     wscript
```

To add a new module to NS-3, we should go through the following steps:

- **Step 1—Create a Skeleton:** NS-3 is integrated with a python program in the source directory that allows us to create a skeleton for a new module. From the src directory, we do the command: *$ ./create-module.py lora* to create the new module, namely *lora*. When it is successful, we obtain the LoRa folder with its directory layout as below:

```
src/
    lora/
         bindings/
         doc/
         examples/
                  wscript
         helper/
         model/
         test/
                  examples-to-run.py
         wscript
```

- **Step 2—Declare Source Files and Public Header Files:** the public header (i.e., files with extension .h) and source code files (i.e., files with extension .cc) for the new module should be specified in the *wscript* file by modifying it with a simple text editor, *gedit* in Ubuntu linux for example.

As an example, after declaring the *lora* module, the *src/lora/wscript* specifies the source code files as below:

```
def build(bld):
    module = bld.create_ns3_module('lora', ['network','mobility'])
    module.source = [
        'model/lora-channel.cc',
        'model/lora-phy-gen.cc',
        'model/lora-mac.cc',
        'model/lora-net-device.cc',
        'model/lora-prop-model.cc',
        'model/mac-lora-gw.cc',
        'model/lora-phy-dual.cc',
        'model/lora-application.cc',
        'model/lora-error-model.cc',
        'model/lora-mac-command.cc',
        'model/lora-mac-header.cc',
        'model/lora-network.cc',
        'model/lora-phy-header.cc',
        'model/lora-noise-model.cc',
        'model/lora-prop-model-ideal.cc',
        'model/lora-test-application.cc',
        'model/lora-address.cc',
        ...
```

The header files defining the public API of our model also be specified in the *wscript* file as below:

```
def build(bld):
    module = bld.create_ns3_module('lora', ['network','mobility'])
    ...
    headers = bld(features='ns3header')
    headers.module = 'lora'
    headers.source = [
        'model/lora-channel.h',
```

```
        'model/lora-phy.h',
        'model/lora-mac.h',
        'model/lora-net-device.h',
        'model/lora-prop-model.h',
        'model/mac-lora-gw.h',
        'model/lora-phy-dual.h',
        'model/lora-application.h',
        'model/lora-error-model.h',
        'model/lora-mac-command.h',
        'model/lora-mac-header.h',
        'model/lora-network.h',
        'model/lora-phy-header.h',
        'model/lora-noise-model.h',
        'model/lora-prop-model-ideal.h',
        'model/lora-test-application.h',
        'model/lora-address.h',
        ...
```

In this way, the API of our model will be published and accessible to users from other modules in NS-3.

- **Step 3—Implement the new module:** until now, we have set everything up for the build system to create code for our new module. We then fall into the process of software development: design, code, compile, run, debug, and repeating each of these steps as needed.

- **Step 4—Declare Tests:** to test our module, we first need to implement test suites, then declare them in our *wscript* file as below:

```
def build(bld):
    module = bld.create_ns3_module('lora', ['network','mobility'])
    ...
    module_test = bld.create_ns3_module_test_library('lora')
    module_test.source = [
        'test/lora-test.cc',
    ]
```

```
...
```

- **Step 5—Declare Examples:** for making scenarios corresponding to our module, it is useful if we can provide several examples. Similar to the test suites, we first need to build the examples, then specify them in *examples/wscript* file as below:

```
def build(bld):
    obj = bld.create_ns3_program('lora-example', ['lora',...])
    obj.source = 'lora-example.cc'
    obj.source = 'lora-energy-example.cc'
    ...
```

Note that the second parameter of the function *create_ns3_program*() is the list of modules that the example being created depends on. Here, we have to include the new module (i.e., *lora*) in the list. We should list only the direct module dependencies, and let the *waf* tool deduce the full dependency tree.

- **Step 6—Examples Run as Tests:** in addition to running test code explicitly, we can use the test framework to run examples to try and catch regressions in them. The script *test/examples-to-run.py* is responsible for handling the calling of NS-3 examples when the test framework turns into running status.

- **Step 7—Python Bindings:** adding Python bindings to our module is not required, and this option is commented out by default in the *create-module.py* script.

```
#! /usr/bin/env python
...
if bld.env.ENABLE_EXAMPLES:
     bld.recurse('examples')

# bld.ns3_python_bindings()
...
```

If we want to include Python bindings (needed only if we write Python programs instead of C++ programs in NS-3), you should uncomment the above option and install the Python API then scan our module to generate new bindings.

- **Step 8—Configure and Build:** in this step, we will get to know how we can configure and build our module. First, we need to ensure that the following tools are available in our Linux or macOS environment:

```
- c++ compiler (clang++ or g++)
- python (python2 version >= 2.7 and python3 version >=3.4)
- git (any recent version)
- tar (any recent version)
- bunzip2 (any recent version)
```

Then, we configure and build the module with three commands as below:

```
$ ./waf clean
$ ./waf configure --enable-tests --enable-examples
$ ./waf build
```

In three commands above, the first one (*./waf clean*) is not very necessary but is a good practice. It will remove the previously built libraries and object files found in directory *build/*.

Then, we can run the unit tests of NS-3 with the following command:

```
$ ./test.py
```

When the test process is terminated, we will get a test report as below:

```
90 of 90 tests passed (90 passed, 0 failed, 0 crashed)
```

This test result is the important information for us to check for failures, crashes, or valgrind errors so as to find out issues in the source code.

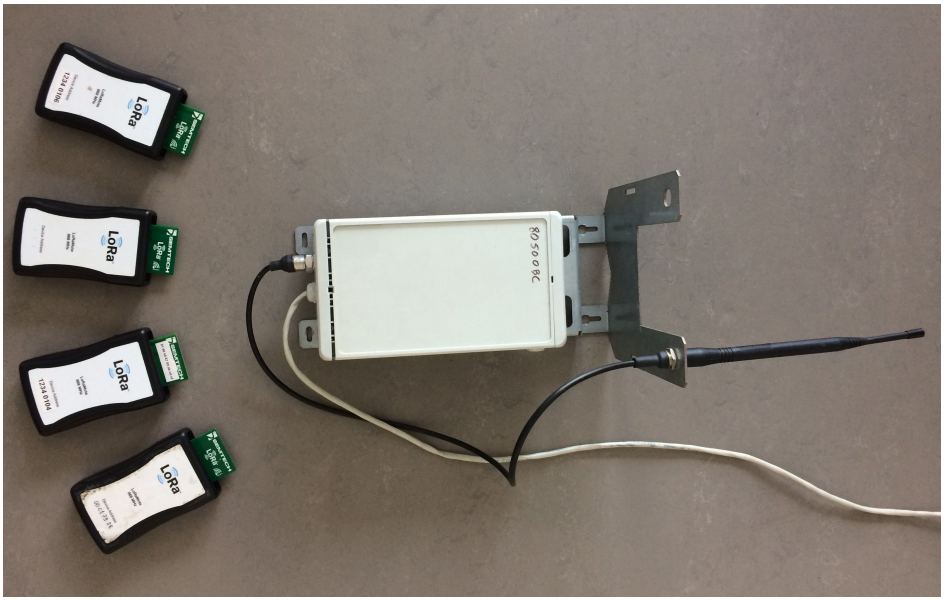

Figure 4.5 – The LoRa testbed with 4 Semtech SX1272LM1BAP end devices and a Kerlink IoT gateway.

4.6 Validation of the LoRa Module

We have first validated the LoRa module on a testbed composed of four Semtech SX1272LM1BAP end devices and a Kerlink IoT Station used as a gateway (see Figure 4.5). We have run several experiments in a real world environment (see Figure 4.6) to measure the packet delivery rate and the packet loss ratio in function of the distance between end devices and the gateway using SF from 7 to 12. Figures 4.7 and 4.8 show the comparison between the measured values and the results of a NS-3 simulation in the same topology. The presence of objects in the environment (e.g., buildings) is the main reason for a small difference between the measured and simulated values.

We have also compared the results of the LoRa module with the measurements reported by Haxhibeqiri et al. [3] for the same simulation parameters. The goal is to validate the capture model in the module because their measurements showed an important impact of the capture effect that lowers the packet drop rate due to collisions. We have used a threshold-based model available on NS-3 for the packet capture effect [63]. We have considered a scenario with the number of devices up to 1000. Figure 4.9 shows the packet loss rate and the collision ratio predicted by the simulation compared to the results by Haxhibeqiri et al. [3].

In addition to our LoRa module, there are at least three different developments of

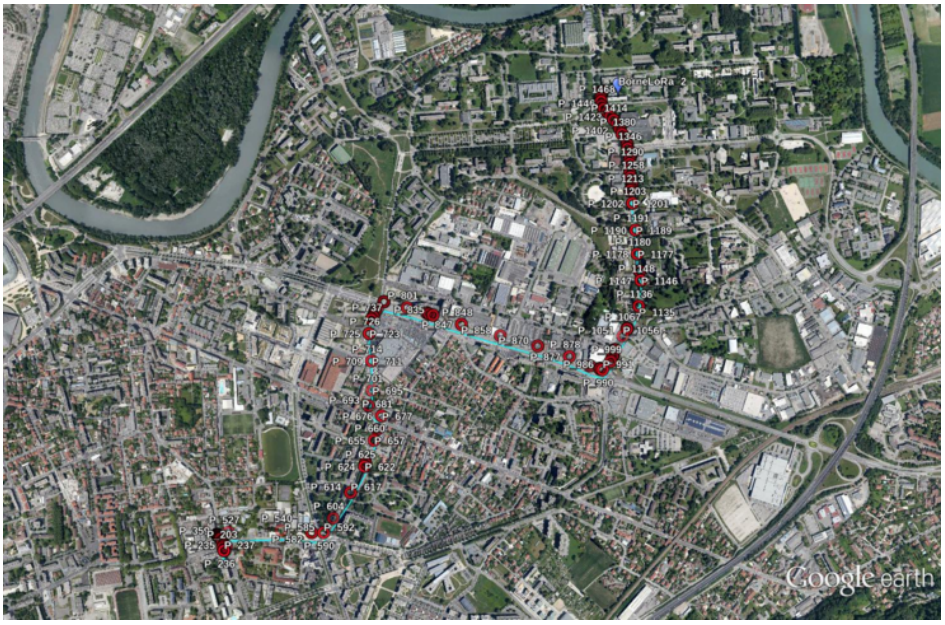

Figure 4.6 – The real environment for deployment the testbed. The distance from end devices to the gateway is increased gradually.

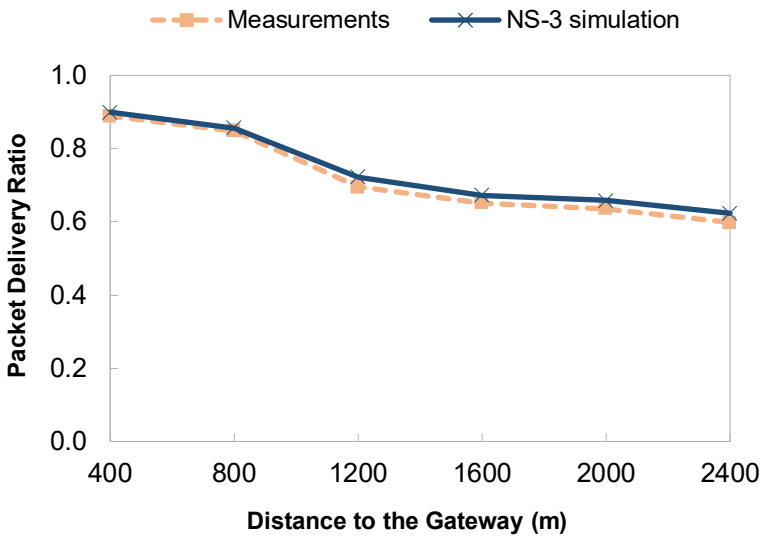

Figure 4.7 – Comparison between measured and simulated values: packet delivery ratio in function of the distance between end devices and the gateway.

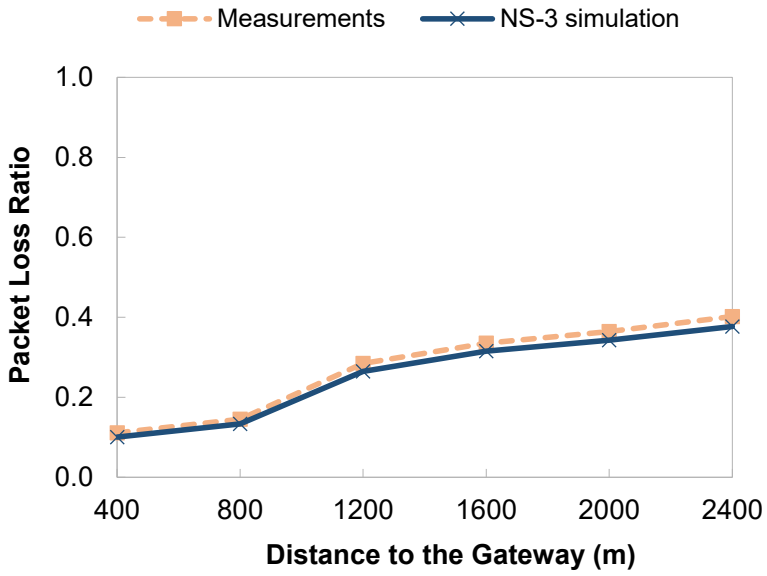

Figure 4.8 – Comparison between measured and simulated values: packet loss ratio in function of the distance between end devices and the gateway.

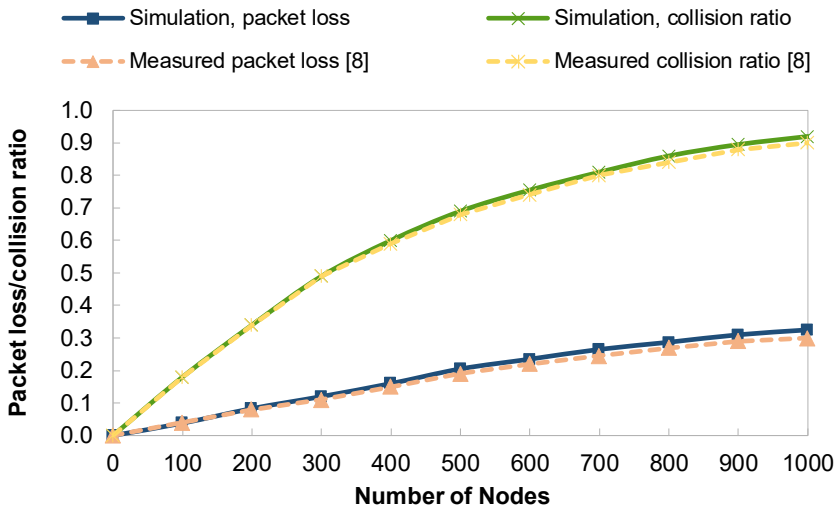

Figure 4.9 – Comparison between simulated values and measurements by Haxhibeqiri et al. [3] showing the impact of the capture effect.

a NS-3 LoRa module [64, 65, 66] published until now.

4.7 Conclusion

This chapter gives the details of the implementation of our LoRa module in NS-3. It comprises modules for LoRa device class A, LoRa Gateway, LoRa channel, and Network Server. The Network Server includes the role of a Join Server as well. We used the module to simulate different scenarios of LoRa networks. At the same time, based on this module, we developed our ideas for LoRa improvement based on the CSMA protocol and a new scheduling scheme called Timemaps, presented in detail in next chapters.

Chapter 5

Improving LoRa Performance with CSMA

Contents

5.1 Introduction

In this chapter, we present two main aspects. First, we overview the principles of ALOHA and CSMA [13]. CSMA consists of testing the channel if it is used by another transmission before attempting to send a packet. The principle, also referred to as "*Listen Before Talk (LBT)*", appears in the ETSI regulations: without LBT, devices need to limit their duty cycles to 0.1% or 1% depending on the sub-band. Thus, if devices apply the CSMA principle, the limitation is released so devices can use higher duty cycles, which contributes to possibly increased throughput and larger network capacity. CSMA-x is another variant of CSMA in which a device listens to the channel for a small of time called CCG (Clear Channel Gap) before attempting a transmission. We use the NS-3 simulator of LoRa to evaluate CSMA and CSMA-x compared to pure LoRaWAN. The enhanced methods result in a much better collision ratio while only slightly increasing energy consumption. For higher loads, CSMA-x even shows

an improvement with respect to LoRaWAN. They enable higher transmission rates by getting rid of the 868 MHz ISM band restrictions.

5.2 ALOHA Protocol

5.2.1 Introduction

The ALOHA access method allows multiple users (senders) to send data packets via a radio link over a common channel to a central station in an uncoordinated manner. The central station (see Figure 5.1) will respond to all senders on another channel. That is, the senders will receive an answer from the central station about the status of their data transmission. If the sender packet is lost due to collisions or signal attenuation, the sender will retransmit the packets some time later [67].

ALOHA protocols are divided into two main types, namely Pure ALOHA and Slotted ALOHA. With pure ALOHA, a device wakes up and sends a packet to the base station at any instant. In slotted ALOHA, the channel is divided into time slots with a fixed size. A device can send a packet to the base station in any slot.

5.2.2 Pure ALOHA

The pure ALOHA channel access method was designed in 1970. Its basic idea is simple: it allows users to transmit whenever they have data to send. The way packets are transmitted on the channel in the ALOHA system is illustrated in the Figure 5.2.

To calculate the throughput of pure ALOHA, we assume that devices in the network generate their data packets at random instants. Therefore, the number of packets in a given time period follows a Poisson distribution. We indicate the time at which a device begins its transmission by t_0, and the time duration needed for a packet to be transmitted by t. Hence, if any other device transmits between $t_0 + t$ and $t_0 + 2t$, it will result in a collision (see Figure 5.3). We consider t as the unit of time, and G as the average offered traffic load per unit of time. In other words, G is the rate of Poisson distribution. Throughput S, as the probability to receive a successful transmission per

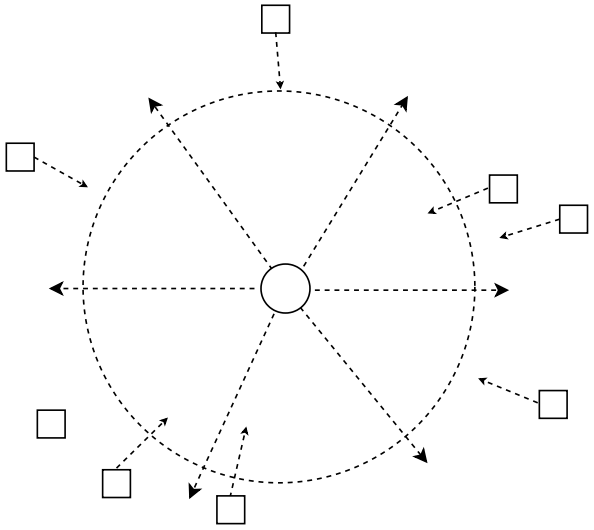

Figure 5.1 – The central station (circle) responds to signals sent by the ALOHA users (squares).

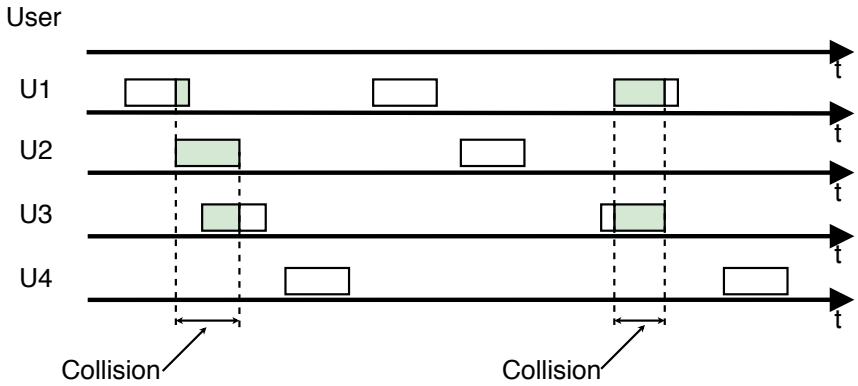

Figure 5.2 – Frames are sent at random instants in pure ALOHA.

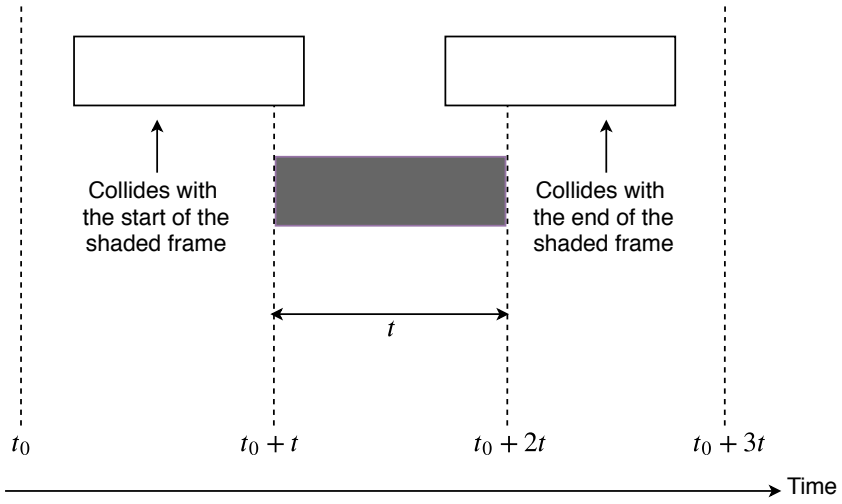

Figure 5.3 – The vulnerable period for the shaded frame.

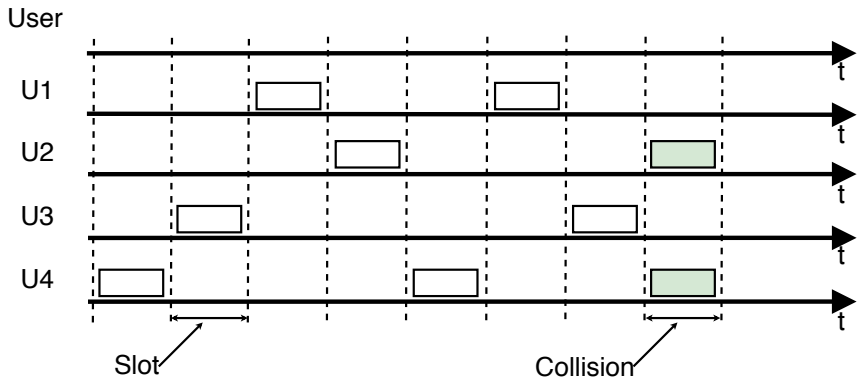

Figure 5.4 – Slotted ALOHA protocol.

unit of time, would be expected as:

$$S = Ge^{-2G} \tag{5.1}$$

which is the product of the offered traffic load per time unit G, and the probability of not existing any other transmission in a specific time unit. The maximum throughput of pure ALOHA is achieved at $G = 0.5$, with $S = 1/2e$, or approximately 0.184 [68].

5.2.3 Slotted ALOHA

Slotted ALOHA was published in 1972 with the ability to double the capacity of the ALOHA system. In this protocol, the time is divided into time slots with a fixed size, each interval corresponding to the transmission of one frame as in Figure 5.4. This approach requires devices to agree on slot boundaries through time synchronization.

In slotted ALOHA, device packet generation is considered to have a Poisson distribution with a traffic load of G. The throughput of this access method will be:

$$S = Ge^{-G} \tag{5.2}$$

The maximum throughput of slotted ALOHA is achieved at $G = 1$ with $S = 1/e$, equivalent to 0.368, double of that of pure ALOHA due to the fact that in slotted ALOHA, collisions can occur only half the number of collisions occurring in pure

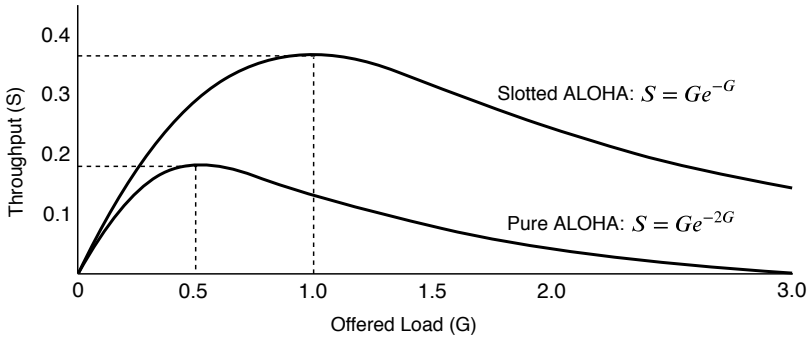

Figure 5.5 – Throughput versus offered traffic for ALOHA systems.

ALOHA [68]. Figure 5.5 shows the throughput under offered traffic for pure ALOHA and slotted ALOHA.

We can see that the throughput of pure ALOHA is half that of slotted ALOHA. However, pure ALOHA does not require synchronization. The ALOHA protocol is very simple and distributed in the sense that the devices operate independently of each other and require a very small amount of feedback information to make their decisions. In addition, the protocol induces very small packet delays when the system is lightly loaded. However, the throughput of both systems is low. Therefore, the question arises whether the throughput of the ALOHA protocol can be improved, while maintaining its desirable features. These considerations lead to the development of CSMA protocols that we discuss in the next section.

5.3 CSMA Protocol

5.3.1 Introduction

The ALOHA protocol is suitable for networks with light channel load. Devices can use the whole channel, send very quickly, and often without (or very little) collision. With a heavier channel load, ALOHA faces the problems of efficiency and stability. Therefore, a number of different signal processing protocols and methods have been invented to reduce these problems. One of these is an improvement of the ALOHA protocol with Carrier Sense Multiple Access (CSMA).

It is often possible for one device to detect what other devicess are doing, and thus

adapt its behaviour accordingly. Protocols in which devices listen for a carrier (i.e., a transmission) and act accordingly are called carrier sense protocols. In this section, we will look at several versions of carrier sense protocols [68].

5.3.2 Persistent CSMA

The first carrier sense protocol that we present in this section is called **1-persistent CSMA**, the simplest CSMA scheme. When a device has data to send, it first listens to the channel to see if anyone else is transmitting at that instant. If the channel is idle, the device sends its data. Otherwise, if the channel is busy, the device just waits until it becomes idle. Then the device transmits a frame. If a collision occurs, the device waits a random amount of time and starts again. The protocol is called 1-persistent because the device transmits with a probability of 1 when it realizes the channel is idle [68].

We expect that this access method avoids collisions (except for the rare case of devices sending simultaneous data packets). However, 1-persistent CSMA does not do that. Indeed, if two devices are ready in the middle of the transmission of the third device, both will wait until the transmission ends, and then both will start transmitting exactly simultaneously, resulting in a collision.

Even so, this protocol has better performance than pure ALOHA because both stations have the decency to desist from interfering with the third station frame. Exactly the same holds for slotted ALOHA.

Despite this, the protocol performs better than pure ALOHA because both devices do not interfere with the transmission of the third device.

The second carrier sense protocol is **p-persistent CSMA**. It applies to channels that are divided into slots. When a device is ready to send, it first senses the channel. If the channel is idle, the device transmits its data with probability p. At the same time, with a probability $q = 1 - p$, the device will delay until the next slot. If that slot is also idle, it either transmits or delays again, also with probabilities p and q. This process is repeated until either its data has been transmitted or another device has started transmitting. In the latter case, the unlucky device works as if there had been a collision (i.e., it waits for a random time and starts again). If the device initially

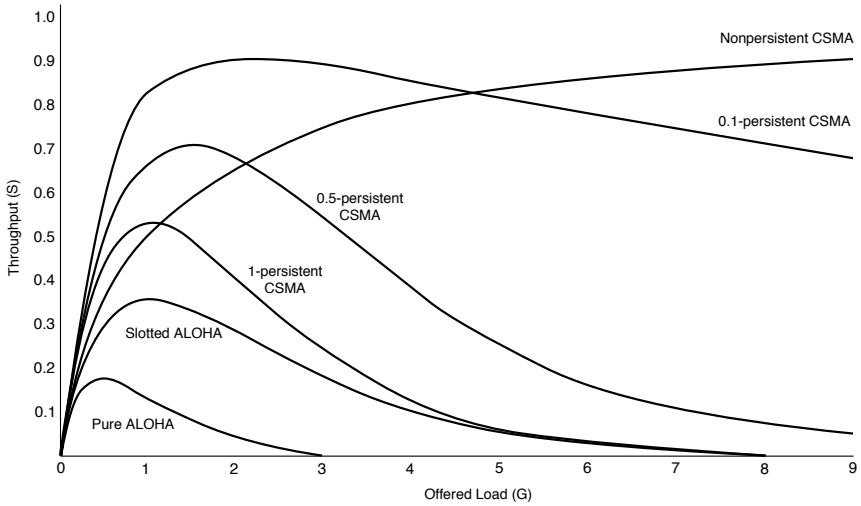

Figure 5.6 – Comparison of the channel utilization versus offered load for CSMA and ALOHA protocols.

senses that the channel is busy, it waits until the next slot and applies the above algorithm.

5.3.3 Nonpersistent CSMA

The third carrier sense protocol is **nonpersistent CSMA**. In this protocol, when a device has its data to send and it senses a busy channel, it waits for a random period of time (without sensing the channel) and repeats the algorithm. Consequently, its rate of collisions is much reduced than that of 1-persistent CSMA. This is because each device waits for a random amount of time before attempting retransmission. The probability that multiple devices will wait for same amount of time is extremely low. So, collisions between contending devices are greatly reduced [68]. We use the idea of nonpersistent CSMA protocol for our first improvement of LoRaWAN performance.

Figure 5.6 shows the channel utilization versus offered traffic for all mentioned CSMA protocols as well as for pure ALOHA and slotted ALOHA.

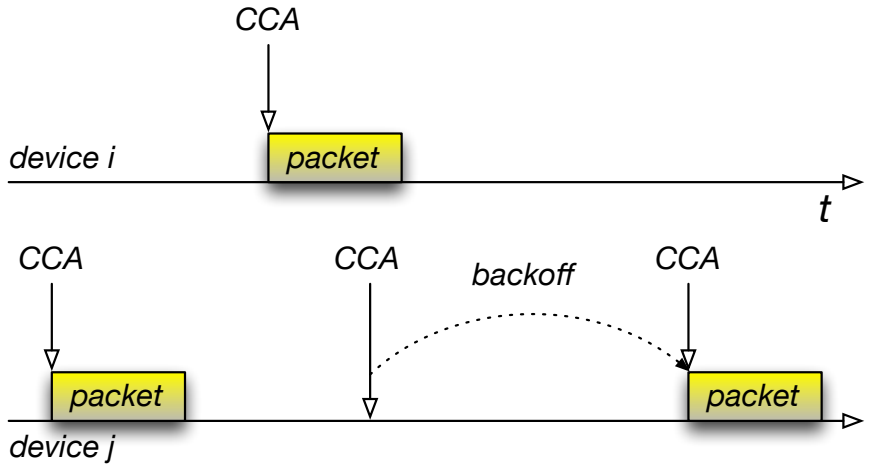

Figure 5.7 – Principle of CSMA: device j sends a packet after a CCA and backs off when channel is busy.

5.4 Improving LoRa with CSMA

In this section, we present the principle of CSMA for LoRaWAN. We assume N contending devices. When end device $i \in N$ has a packet to send, it randomly chooses communication channel c_i. It performs CCA (Clear Channel Assessment) to test if there is an ongoing transmission on the channel. Only when the channel is clear, the device starts its transmission, otherwise, it backs off— it goes to sleep for a random interval of time and attempts a transmission later on. The random interval is equal to k slots of 1 s, where $k \in [0, 2^n - 1]$ for the n^{th} transmission attempt (the maximum value of n is set to 3). Figure 5.7 illustrates the principle.

Another variant of CSMA that we call CSMA-x is to listen to the channel for a small interval of CCG time before attempting a transmission. When the device detects a transmission during this interval, it backs off as in the basic CSMA. Figure 5.8 illustrates the principle.

To investigate the energy consumption at each node and in the whole network, we use Lithium-ion batteries as the type of energy sources and assign a device energy model to each end device. There are interactions between the energy source and the device energy model: the model consumes the energy from the source and the source notifies the model when its energy is completely drained.

The parameters of energy consumption in each state come from the datasheet of

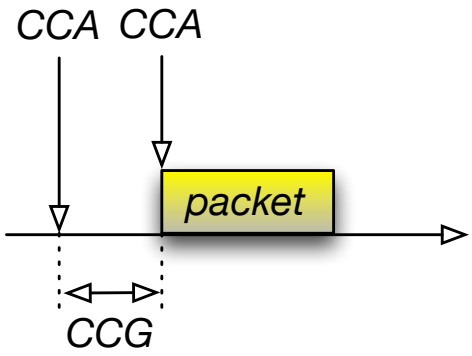

Figure 5.8 – Principle of CSMA-x: device j sends a packet after a CCA and a CCG interval.

Table 5.1 – Power consumption in LoRa

P_{tx}	419.6 mW
P_{rx}	44.06 mW
P_s	4.32 μW

LoRa SX1272 [69] and the Low Energy Consumption Design for SX1272/3/6/7/8 LoRa Modem [70] (see Table 5.1).

5.5 Simulation Results

In this section, we describe the scenario used in simulations and their results.

5.5.1 Simulation Scenario and Settings

We consider a scenario with one gateway and a number of end devices up to 10000 nodes. The simulation time is limited to 20000 seconds. The positions of end devices are randomly distributed around the gateway in the area of 10000 m x 10000 m. End devices send unconfirmed data frames. We simulate LoRaWAN, CSMA, and CSMA-1 (CSMA-x with the interval of 1 CAD (Channel Activity Detection), i.e., 61 ms). Table 5.1 presents other parameters.

Table 5.2 – Simulation parameters

Voltage	3.3 V
Frequency Band	868 MHz
Code Rate	4/5
Bandwidth	125 kHz
Duty Cycle	1%
Output Power	20 dBm
Payload Length	10 bytes
Preamble Length	12 symbols
Number of channels	3
Spreading Factor	SF7-SF12

5.5.2 Results

We have evaluated CSMA and CSMA-1 and compared their performance with LoRaWAN using NS-3 with respect to packet delivery ratio, collision ratio, and energy consumption. We have simulated a network with an increasing number of nodes from 1 up to 10000. If a packet is being received by the receiver and it is receiving the preamble of another packet, we consider this situation as the capture effect: if the value of Signal to Interference Ratio (SIR) of the new packet is above a specific threshold, then the current packet is dropped and the receiver locks on the new incoming packet.

Figure 5.9 presents the packet delivery rate for LoRaWAN, CSMA, and CSMA-1 (the ratio of the number of received packets by the gateway to the number of all packets transmitted by end devices). We can observe a much better rate for CSMA compared to LoRa. The figure also shows better scalability of CSMA—it obtains the packet delivery rate greater than 90% for more than 4000 devices. The rate for CSMA-1 is even higher than that of CSMA when the number of devices rises up to 1500 devices and more.

Figure 5.10 presents the collision ratio for LoRaWAN and CSMA: the number of dropped packets because of collisions to the number of all transmitted packets. We can notice that the ratio rapidly increases for LoRaWAN with the number of contending devices. The increase of CSMA is much more moderate because devices send much less packets involved in collisions.

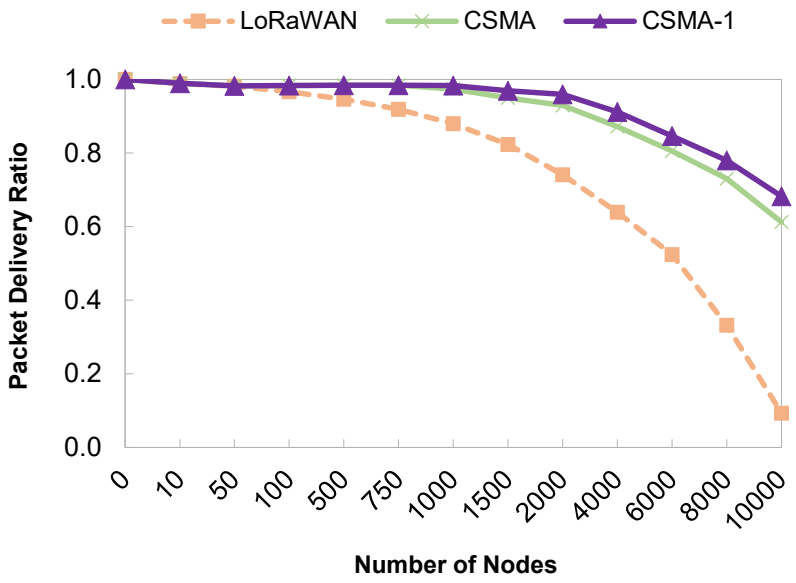

Figure 5.9 – Packet delivery ratio in the whole network under LoRaWAN, CSMA, and CSMA-1.

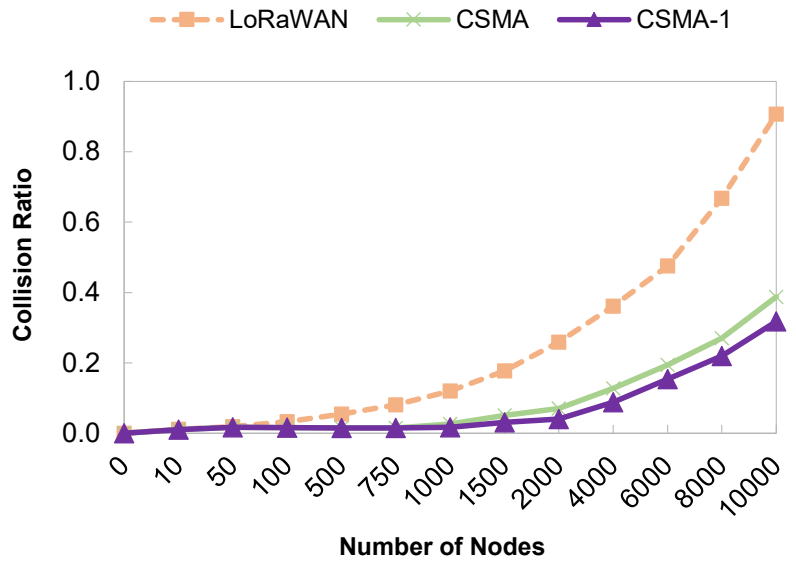

Figure 5.10 – Collision ratio in the whole network under LoRaWAN, CSMA, and CSMA-1.

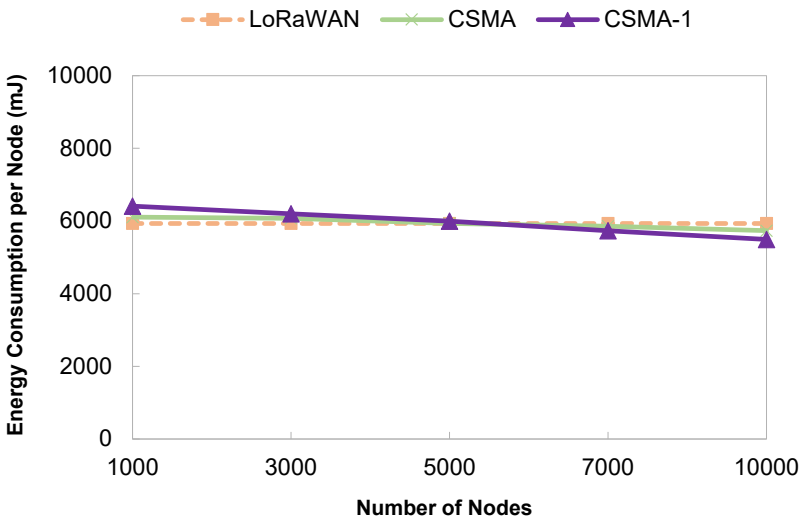

Figure 5.11 – Energy consumption per node under LoRaWAN, CSMA, and CSMA-1.

Figure 5.11 presents the energy consumption of LoRaWAN, CSMA, and CSMA-1. In range of 0 – 5000 devices, the total consumed energy for CSMA is higher than that of LoRaWAN because of the interval before transmission during which a device is awake. The trend is inversed for the range of 5000 – 10000: the energy consumption of CSMA is lower than that of LoRaWAN. For the large number of contending devices, there are more ongoing transmissions so the first CCA detects a transmission and the device backs off.

5.6 Conclusion

In this chapter, we present a performance improvement to the LoRa access method based on CSMA to lower the collision ratio. We have used the NS-3 simulator to evaluate two schemes: CSMA and CSMA-1. The simulation results show that CSMA considerably lowers the collision ratio while only slightly increasing energy consumption. We also observe that CSMA-1 presents lower energy consumption than LoRaWAN for a large number of devices. Another advantage of CSMA consists of increased throughput and larger network capacity because the ETSI restrictions on the duty cycle do not longer apply.

In parallel to our work, Pham [71, 72] proposed a CSMA protocol adapted to LoRa to reduce collisions in LoRa networks. His proposal builds on the 802.11 access

method: a node wakes up and performs periodic CAD during a DIFS (Distributed Coordinated Function Interframe Space) interval to detect on-going transmissions. If CAD is unsuccessful, a node will wait the ToA (Time on Air) of the longest LoRa packet until the next attempt.

In our mechanism, a node performs only 3 CAD (i.e., 3×61 ms) to limit energy consumption and in case of unsuccessful CAD, the node will wait a random interval of time whose size increases with the number of transmission attempts.

Chapter 6

Timemaps for Improving Performance of LoRaWAN

Contents

6.1 Introduction

We start from the observation that the ALOHA type of access leads to a disordered behavior of devices sending their packets at potentially every instant. Enforcing some ordering in transmissions would benefit to all devices. At the same time, Gateways and Network Servers are the elements through which all traffic passes so they have all the information about the pattern of arriving packets and the transmission parameters used by devices.

We propose to make the access method more ordered with *Timemaps*, temporal maps of transmissions built by Gateways to schedule transmissions and avoid collisions. The idea is the following: a device when performing a Join operation includes its traffic description in the request giving the information on the *size* of data that it will generate and *periodicity*, the interval between transmissions. These parameters are specific to an IoT application—what data the application generates and how often. Based on the traffic descriptions from all devices, the Gateway constructs a schedule for channel access that avoids collisions.

When responding to the Join request, the Gateway includes the information on the temporal position of the device in the schedule, the SF to use for transmissions based on the measured Signal to Noise Ratio (SNR) at the Gateway, and the channels to use. Devices set up waking instants to their temporal position in the schedule and use recommended SF. This operation relies on the common notion of time at devices while the clocks of LoRa devices may derive during long periods when they go to sleep to save energy. To mitigate the effect of clock drift, we propose two mechanisms: i) devices need to wake up at least every maximal interval and time synchronize with the Gateway, ii) devices estimate the clock drift based on the synchronization to compensate for the eventual time difference. The interval between the time synchronization can be shorter than the periodicity of sending application data but much longer than the interval between beacons for Class B devices (128 s) so that energy consumption is still limited. Periodic synchronization is also a means for providing devices with new configuration parameters based on the current state of transmission conditions measured by the Gateway.

Timemaps make transmissions more regular and may significantly lower the probability of collisions even if the method cannot eliminate them completely due to the clock drift.

We evaluate the performance of the proposed scheme with simulations in NS-3 for two cases: i) ideal time synchronization of devices and ii) realistic clock drift of 40 ppm.

6.2 Improving LoRaWAN with Timemaps

In this section, we first introduce some assumptions and then, we present the principles of packet transmission scheduling based on Timemaps.

6.2.1 Assumptions

We assume that a device performs Over The Air Activation (OTAA) by sending a signed Join Request frame providing the required information for device authentication ($JoinEUI$, $DevEUI$, $DevNonce$). If the Join Server accepts the device, it derives the

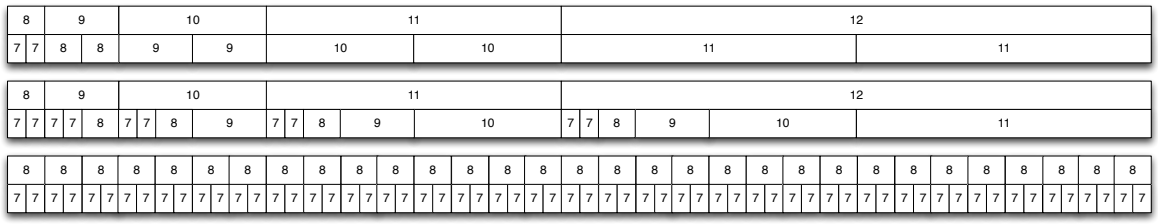

Figure 6.1 – Timemaps for scheduling with transmission slots for different spreading factors.

session keys and sends a Join Accept with $AppNonce$, $NetID$, $DevAddr$ along with other information such as downlink parameters, $RxDelay$ (the delay for waking up for receiving an ACK), and a list of channels to use. Based on $AppNonce$, $NetID$, $DevAddr$, the device derives the session keys and can start sending application traffic.

We use the regular procedure of OTAA as in LoRa, but we propose to add more information to the join exchange—the description of traffic that will be generated by the device. The traffic description of device i includes the following information:

- *size* N_i of application data to generate,
- *periodicity* T_i, the tentative interval between transmissions.

The receiving Gateway measures SNR_i for device i and based on the traffic descriptions of all devices that have joined the network, it constructs *Timemaps* for each channel—schedules of all tentative transmissions built to avoid collisions.

6.2.2 Scheduling transmissions based on Timemaps

The scheduling idea is to divide the airtime into time slots and propose the initial transmission instant to Lora devices. The scheduling strategy needs to address the following requirements:

- we need to distribute slots as uniformly as possible across the airtime so that if a device wants to send data at time T_i, the Gateway can allocate a slot close to T_i. For instance, if a device wants to send data every day at noon, the Gateway needs to allocate a slot close to noon.

- provide for a flexible number of slots for each SF so that a Gateway can easily satisfy the demand devices having specific transmission conditions.

- make possible overlapping of slots of different SF (no overlapping of slots of the same SF and no overlapping of more than two transmissions). As devices usually allocate SF in function of SNR that decreases with the distance from the Gateway, transmissions of different SF will be practically orthogonal according to Table 3.10.

- specify a simple algorithm for incrementally allocate slots to requesting devices by computing the position of a given slot (easy calculation of the allocation and not a complex lookup).

Based on the received traffic descriptions, a Gateway constructs a Timemap for scheduling transmissions. We propose the scheduling strategy presented below.

Timemaps and SF$_j$ slots. We define a Timemap pattern presented in the first row of Figure 6.1. The size of a slot for SF$_j$ is τ_j and the unit of allocation is the maximal transmission time for SF7: $\tau_7 = 102.7$ ms. We define the relationships between τ_j and τ_{j-1} as the double of the slot for the lower SF:

$$\tau_j = 2\tau_{j-1}, j = 8, ..., 12.$$

For some SF, the duration of the slot is greater than the actual airtime, but this simple recurrence relation allows for making the time diagram simple and computable.

We construct the Timemap by filling up the airtime with SF8 slot overlapping with two SF7 slots, followed by a SF9 slot overlapping two SF8 slots, etc. In this way, we obtain the basic schedule with a balanced number of slots for each SF (2 SF7, 3 SF8, 3 SF9, 3 SF10, 3 SF11, and 1 SF12 in the figure).

If there is no slot for SF$_j$, the Gateway can divide a slot for SF$_{j+1}$ into two slots for SF$_j$, so for instance if we need more SF7 slots, we can divide one SF8 slot into two SF7 slots, which results in the Timemaps presented in the second row of Figure 6.1 (10 SF7, 5 SF8, 4 SF9, 3 SF10, 2 SF11, and 1 SF12 in the figure). More SF7 slots means that we can support more devices close to the Gateway that benefit from good transmission conditions and low energy consumption. The limit of this slot division

process results in a Timemap with many SF7 and SF8 slots (62 SF7, 31 SF8, see the third row in Figure 6.1).

The Gateway maintains the Timemap pattern for all three channels available in the 868 MHz band. When device i sends a Join request, the Gateway finds the slot closest to the instant at which a device wants to transmit its data and makes periodicity T_i correspond to the interval between two SF_j slots if the device SNR allows using this SF value.

Compensating clock drift. Timemaps are effective if devices have the common notion of time but usually, their clocks may derive during long periods when devices go to sleep to save energy. More formally, we assume that devices have an imperfect clock $C(t)$ with a given bounded drift $\triangle$ that satisfies $|\frac{dC(t)}{dt} - 1| \leq \triangle$. A typical value of $\triangle$ for crystal clocks is 40 ppm. We also assume that Gateways have high resolution stable clock also needed for geo-localization. The problem of clock drift affects transmissions, reception of data, and energy consumption [2].

We propose to mitigate the effect of clock drift in two ways. First, we require that devices wake up at least every T_s interval, send a *DeviceTimeReq* command to the Gateway to receive the *DeviceTimeAns* command [1] with a timestamp. T_s, the interval between time synchronization can be shorter than the periodicity of sending application data but we can set it to a value much longer than the interval between beacons for Class B devices (128 s) so that energy consumption related to time synchronization remains limited. Periodic synchronization is also a means for providing devices with new configuration parameters based on the current state of transmission conditions measured by the Gateway.

Second, the device estimates the clock drift based on the received timestamp in a similar way we proposed for 802.15.4 TSCH networks [73]. If we represent the device clock as $C(t) = Dt$, the idea is to estimate $\hat{D}$ based on the timestamp received from the Gateway:

$$\hat{D} = T_s/T_s^{'} \tag{6.1}$$

where T_s and $T_s^{'}$ are the intervals between synchronization measured by the Gateway (assuming perfect clock) and the device, respectively. The device adjust its clock based on the relation $C(t) = \hat{D}t$ to compute the next instant of waking up corresponding to

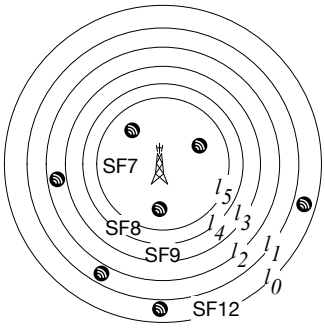

Figure 6.2 – Annuli of SF configurations around a gateway

the scheduled time slot.

Finally, we precede the slots in Timemaps with some time guards to mitigate the influence of imperfect estimation of clock drift $\hat{D}$. Note that Figure 6.1 does not represent any time guards for greater legibility.

6.3 Simulation Evaluation

In this section, we describe the considered configurations of LoRa networks, scenarios, and present the simulation results.

6.3.1 Network Configurations

We consider two cell configurations [44]: i) *equidistant SF boundaries*—devices choose SF (so the data rate) based on the distance to the Gateway and ii) *SNR-based SF boundaries*—the choice of SF depends on SNR measured at the Gateway.

The configurations correspond to the view presented in Figure 6.2 with the configurations differing in the values of l_j, the range of Gateway coverage: l_j is the distance to the farthest device that uses SF$\{12-j\}$ and so data rate DRj. l_0 is the maximum transmission range.

In the first configuration, we assume that devices are homogeneously scattered on the plane with spatial density ρ so the number of nodes using SF$\{12-j\}$ and data rate DRj is proportional to the surface of the annulus between l_{j+1} and l_j around the

Table 6.1 – Equidistant SF boundaries [km], S [km^2]: surface proportional to the number of devices (constant node density ρ for all annuli).

SF	SF7	SF8	SF9	SF10	SF11	SF12
l_j	l_5	l_4	l_3	l_2	l_1	l_0
SF boundary [km]	1	2	3	4	5	6
S/π [km^2]	1	3	5	7	9	11

Table 6.2 – SNR-based SF boundaries [km] [44], P_s is the success probability due to attenuation, fading, thermal noise. S [km^2]: surface proportional to the number of devices. Node density in an annulus based on the inverse-square law.

SF	SF7	SF8	SF9	SF10	SF11	SF12
l_j	l_5	l_4	l_3	l_2	l_1	l_0
$P_s = 90\%$	2.23	2.68	3.23	3.89	4.54	5.30
S/π [km^2]	4.96	2.23	3.24	4.69	5.49	7.47
$\rho(l_j)$	1	0.69	0.48	0.33	0.24	0.13
$S/\pi \times \rho(l_j)$	4.96	1.54	1.54	1.54	1.32	1.32

Gateway.

Table 6.1 presents the values of l_j for the equidistant SF configuration, S being the surface of a disk or annuli proportional to the number of devices in the disk or in the annulus. Similarly, Table 6.2 presents the values of l_j for the SNR-based SF configuration.

Assuming constant node density ρ for all annuli is not realistic because using higher SF results in increased energy consumption, which will discourge the placement of nodes far from the Gateway. A more realistic model for the spatial distribution of nodes is based on the inverse-square law for node density:

$$\frac{\rho(l_j)}{\rho(l_{j-1})} = \frac{l_{j-1}^2}{l_j^2} \tag{6.2}$$

Table 6.3 – Number of nodes in each annulus in homogeneous density (constant node density ρ for all annuli).

SF	SF7	SF8	SF9	SF10	SF11	SF12
l_j	l_5	l_4	l_3	l_2	l_1	l_0
S/π [km^2]	1	3	5	7	9	11
$n = 1000$	28	83	139	194	250	306
$n = 2000$	56	167	278	389	500	610
$n = 3000$	83	250	417	583	750	917
$n = 4000$	111	333	556	778	1000	1222
$n = 5000$	139	417	694	972	1250	1528

Table 6.2 also presents node density $\rho(l_j)$ for each annuli based on this relation and the surface proportional to the number of devices. We can observe that such a distribution of nodes favors devices close to the Gateway with a higher number of devices using SF7 and results in a lower number of devices with SF11 and SF12.

6.3.2 Scenarios and Settings

The network is composed of one Gateway and a number of devices varying up to 5000 nodes. We simulate the distribution of nodes both for homogeneous and inhomogeneous density. Table 6.3 and 6.4 present the number of nodes in each annulus for the both cases, respectively. We choose the sufficient number of time slots to support the corresponding number of devices in each annulus, for instance in the SNR-based SF configuration, for each SF12 slot there should be 5 slots of SF7 (see Table 6.4).

We assume that devices send Unconfirmed Data frames. The simulation time is 43200 seconds (i.e., 12 hours). Each device generates 30 data packets of 59 B on the average during the simulation time. We simulate LoRaWAN, LoRaWAN with CSMA, Timemaps for the ideal clock case (IdealTimemaps), and Timemaps for the typical clock drift of 40 ppm (DriftTimemaps). We set $T_s = 60$ minutes, the time interval between two synchronization operations of a device and the Gateway. We precede each slot in Timemaps with a time guard of 10 ms. Table 6.5 presents other simulation parameters.

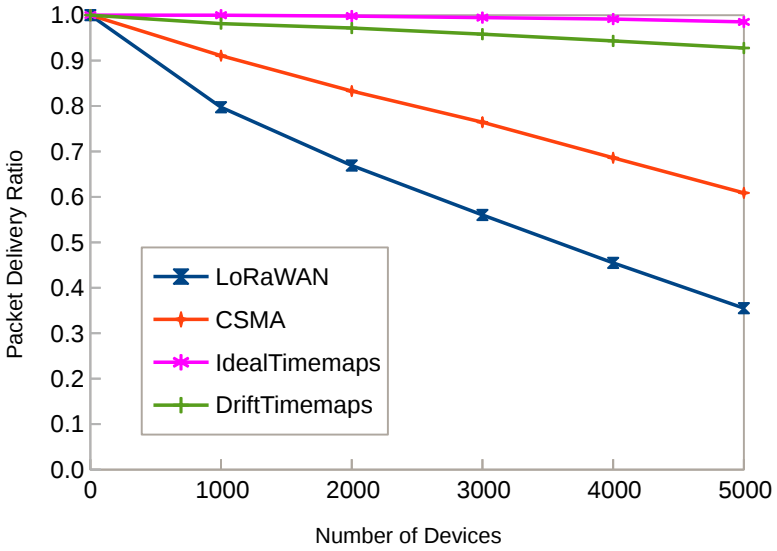

Figure 6.3 – PDR of LoRaWAN, CSMA, IdealTimemaps, and DriftTimemaps for homogeneous density.

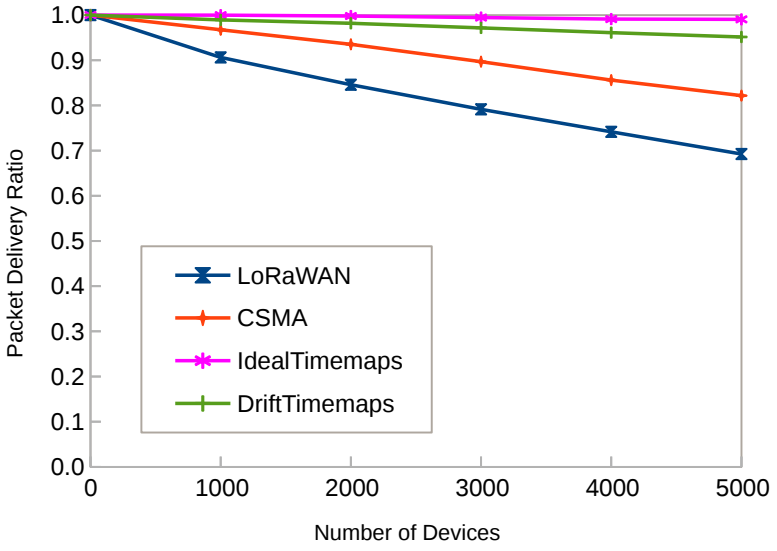

Figure 6.4 – PDR of LoRaWAN, CSMA, IdealTimemaps, and DriftTimemaps for inhomogeneous density.

6.3.3 Performance comparisons

We compare the performance of IdealTimemaps and DriftTimemaps with both LoRaWAN and CSMA in terms of Packet Delivery Ratio (PDR), Collision Ratio and Energy Consumption. Figure 6.3 presents PDR of the access methods for homogeneous

Table 6.4 – Number of nodes in each annulus in inhomogeneous density (node density in an annulus based on the inverse-square law).

SF	SF7	SF8	SF9	SF10	SF11	SF12
l_j	l_5	l_4	l_3	l_2	l_1	l_0
S/π [km^2]	4.96	2.23	3.24	4.69	5.49	7.47
$\rho(l_j)$	1	0.69	0.48	0.33	0.24	0.13
$S/\pi \times \rho(l_j)$	4.96	1.54	1.54	1.54	1.32	1.32
$n = 1000$	417	129	131	130	111	82
$n = 2000$	834	259	262	260	222	163
$n = 3000$	1251	388	392	390	333	246
$n = 4000$	1669	518	523	521	443	326
$n = 5000$	2086	647	654	651	554	408

Table 6.5 – Simulation parameters

Voltage	3.3 V
Frequency band	868 MHz
Coding rate	4/5
Bandwidth	125 kHz
Duty cycle	1%
Max transmit power	14 dBm
Number of channels	3
Spreading factor	SF7-SF12

density. We can observe better PDR of IdealTimemaps and DriftTimemaps compared to LoRaWAN and CSMA. Moreover, the ratio rapidly decreases for LoRaWAN. The decrease of IdealTimemaps and DriftTimemaps is more moderate because the Gateway schedules transmissions to avoid collisions. IdealTimemaps has the highest PDR as it operates under the assumption of ideal clocks. DriftTimemaps also achieves high PDR: about 93% for 5000 nodes.

Figure 6.4 presents PDR of LoRaWAN, CSMA, IdealTimemaps and DriftTimemaps for inhomogeneous density. Compared to Figure 6.3, PDR of all access methods for inhomogeneous density is higher than in the case of homogeneous density. DriftTimemaps

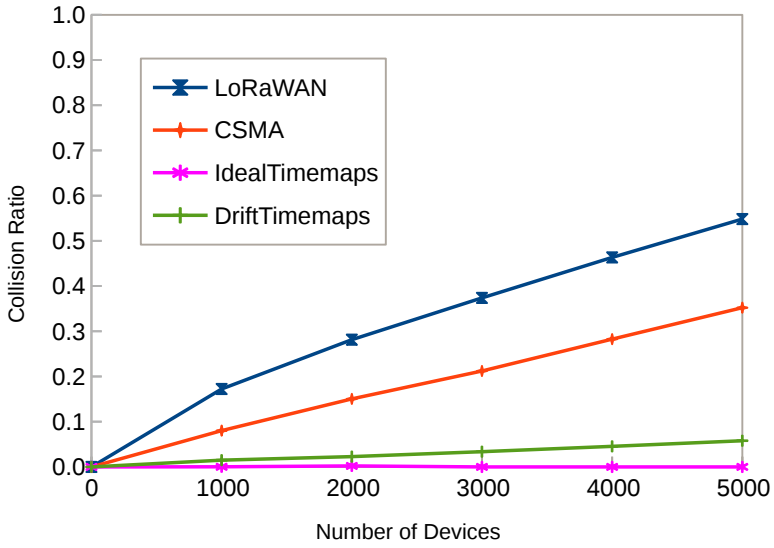

Figure 6.5 – Collision ratio in the whole network under LoRaWAN, CSMA, IdealTimemaps, and DriftTimemaps for homogeneous density.

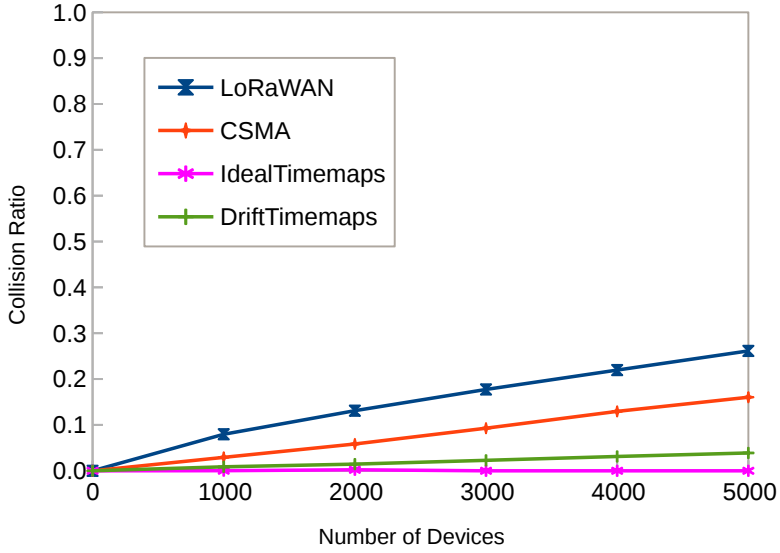

Figure 6.6 – Collision ratio in the whole network under LoRaWAN, CSMA, IdealTimemaps, and DriftTimemaps for inhomogeneous density.

in this context achieves PDR of about 95% for 5000 nodes.

Figures 6.5 and 6.6 show the collision ratio for LoRaWAN, CSMA, IdealTimemaps and DriftTimemaps in the context of both homogeneous and inhomogeneous density. We can notice that the ratio rapidly increases for LoRaWAN with the number of contending devices. The ratio increase for DriftTimemaps remains limited because

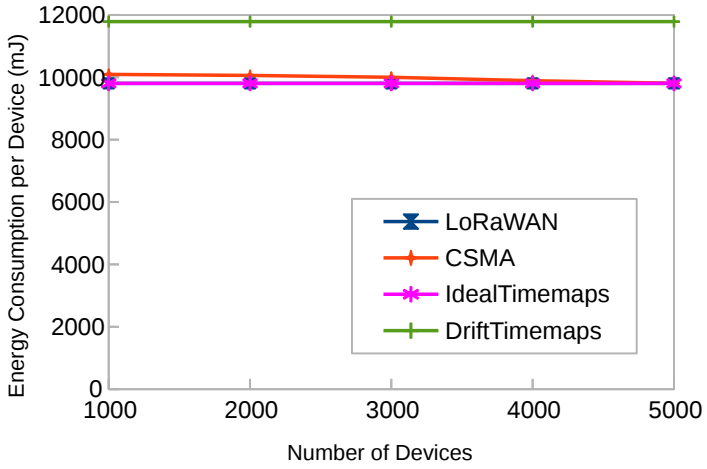

Figure 6.7 – Energy consumption per node of LoRaWAN, CSMA, IdealTimemaps, and DriftTimemaps for homogeneous density.

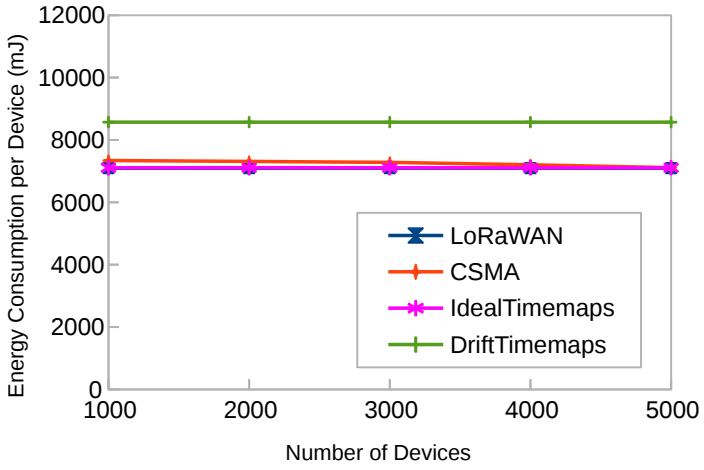

Figure 6.8 – Energy consumption per node of LoRaWAN, CSMA, IdealTimemaps, and DriftTimemaps for inhomogeneous density.

of scheduled transmissions. Collisions under DriftTimemaps mainly happen between data packets and packets for time synchronization. For IdealTimemaps, there are no collisions observed during the simulation.

Figures 6.7 and 6.8 show the energy consumption for LoRaWAN, CSMA, IdealTimemaps and DriftTimemaps for homogeneous and inhomogeneous density, respectively. DriftTimemaps consumes more energy than LoRaWAN because of synchronization operations and guard intervals to compensate for clock drift. Energy consumption for IdealTimemaps is equivalent to that of LoRaWAN. Moreover, for inhomogeneous

density, nodes consume less energy than for homogeneous density because the number of nodes under SF7 is higher than that of SF12.

6.4 Conclusion

This chapter presents Timemaps, a new access method for improving the performance of LoRa. The idea is to build a temporal map of all transmissions of IoT devices by a Gateway and distribute the schedule during the Join procedure. Schedule admits overlapping transmissions of different SFs to increase the overall capacity of the network.

We have used the NS-3 simulator to evaluate our proposal in both cases of perfect clocks and clocks with a drift as well as for homogeneous and inhomogeneous node density. The simulation takes into account quasi-orthogonality of transmissions with different spreading factors and the capture effect. The results show that Timemaps benefits from remarkably higher PDR and a considerably lower collision ratio compared to LoRaWAN along with slightly increased energy consumption.

Unlike previous approaches proposed in the literature, Timemaps takes multiple SFs into account, accepts an unknown number of nodes, and admits overlapping transmissions of different SFs to build the schedule of all transmissions of IoT devices.

Chapter 7

Conclusions and Perspectives

Contents

In this final chapter, we first summarize the main contributions of our dissertation and then outline future research directions for our work.

7.1 Contributions

The first contribution is the LoRa module in NS-3. We have developed a module for NS-3 that simulates the LoRa behavior. The LoRa module consists of four important parts, namely, LoRa Device, LoRa Gateway, LoRa Channel, and Network Server. For LoRa Channel, we use the Spectrum Channel providing support for modeling the frequency-dependent aspects of communications in NS-3. We use the energy framework implemented in NS-3 by Wu et al. [59] to estimate energy consumption at a battery powered node or in the whole network. To validate the module, we have compared its results with measurements on both a real-world testbed and the measured values reported in other work [3].

The second contribution is an improvement of the performance of LoRa devices by CSMA. We have used the NS-3 simulator to evaluate CSMA and CSMA-1, the proposed enhanced access methods that lower the collision ratio. The simulation results show that CSMA considerably lowers the collision ratio while only slightly increasing energy consumption. We also observe that CSMA-1 presents lower energy consumption than LoRaWAN for a large number of devices.

The third contribution is Timemaps, a new access method for improving the performance of LoRa. The idea is to build a temporal map of all transmissions of IoT devices by a Gateway and distribute the schedule during a join. Schedules admit overlapping transmissions of different SF and as devices usually allocate SF in function of the increasing distance from the Gateway, transmissions will be in fact orthogonal, which leads to increased overall capacity of the network. We have used the NS-3 simulator to evaluate our proposal in both cases of perfect clocks and clocks with a drift. The simulation takes into account quasi-orthogonality of transmissions with different spreading factors and the capture effect. The results show that Timemaps benefits from remarkably higher PDR and considerably lower the collision ratio compared to LoRaWAN along with moderately increased energy consumption.

7.2 Perspectives

Our LoRa module in NS-3 just supports LoRa devices under class A. We can enhance the module to support LoRa devices under classes B and C. In addition, the Adaptive Data Rate (ADR) mechanism for optimizing data rates, airtime and energy consumption in the network can be developed to contribute to the LoRa module that better corresponds to the requirements of the LoRaWAN 1.1 specification.

The proposed enhanced access methods—CSMA and Timemaps benefit from remarkably higher PDR and a considerably lower collision ratio compared to LoRaWAN along with slightly increased energy consumption. However, it is assumed that devices send unconfirmed data frames. We can evaluate our proposals in terms of devices sending confirmed data frames. For Timemaps, we can also improve this scheme to adapt to event-based transmissions.

To evaluate Timemaps and CSMA for LoRaWAN, we have used scenarios with just one Gateway. However, it would be interesting to evaluate the protocol in scenarios with multiple Gateways and to compare the results with the previous scenarios. The idea of dense deployments of gateways can also be exploited at another level. When the signal from an IoT device is received by several Gateways, they can exploit interference through Coordinated MultiPoint/Joint Transmission (CoMP-JT): an Edge Server can process the signal received by several Gateways and decode packets even when decoding is impossible at each single Gateway. In this way, we can lower the transmission power

of IoT devices, the aspect especially important for low power nodes.

Our proposals for improving the performance of LoRaWAN based on CSMA and Timemaps have been evaluated by NS-3 simulations. However, it is better to evaluate it on the testbed and real-world environment.

Chapter 8

Publications

- T. To and A. Duda. Simulation of LoRa in NS-3: Improving LoRa Performance with CSMA. In *2018 IEEE International Conference on Communications (ICC)*, pages 1-7, Kansas City, USA, 2018.
- T. To and A. Duda. Timemaps for Improving Performance of LoRaWAN. In *2020 IEEE International Conference on Communications (ICC)*, pages 1-7, Dublin, Ireland, 2020.

Bibliography

[1] N. Sornin and A. Yegin. LoRaWAN Specification v1.1. https://lora-alliance.org/resource-hub/lorawantm-specification-v11, 2017. 1, 2, 13, 23, 27, 38, 55, 56, 89

[2] Elodie Morin, Mickael Maman, Roberto Guizzetti, and Andrzej Duda. Comparison of the Device Lifetime in Wireless Networks for the Internet of Things. *IEEE Access*, 5:7097–7114, 2017. 2, 7, 89

[3] Jetmir Haxhibeqiri, Floris Van Den Abeele, Ingrid Moerman, and Jeroen Hoebeke. LoRa Scalability: A Simulation Model Based on Interference Measurements. *Sensors*, 17(6):1193, 2017. xii, 2, 3, 4, 39, 41, 65, 67, 99

[4] Congduc Pham, Ahcène Bounceur, Laurent Clavier, Umber Noreen, and Muhammad Ehsan. Investigating and Experimenting Interference Mitigation by Capture Effect in LoRa Networks. In *Proceedings of the 3rd International Conference on Future Networks and Distributed Systems*, ICFNDS '19. ACM, 2019. 2

[5] Orestis Georgiou and Usman Raza. Low Power Wide Area Network Analysis: Can LoRa Scale? *IEEE Wireless Commun. Letters*, 6(2):162–165, 2017. 2, 42

[6] Aloÿs Augustin, Jiazi Yi, Thomas H. Clausen, and William Mark Townsley. A Study of LoRa: Long Range & Low Power Networks for the Internet of Things. *Sensors*, 16(9):1466, 2016. 2, 40

[7] Ferran Adelantado, Xavier Vilajosana, Pere Tuset-Peiró, Borja Martínez, Joan Melià-Seguí, and Thomas Watteyne. Understanding the Limits of LoRaWAN. *IEEE Communications Magazine*, 55(9):34–40, 2017. 2, 40

[8] Brecht Reynders, Wannes Meert, and Sofie Pollin. Range and Coexistence Analysis of Long Range Unlicensed Communication. In *23rd International Conference on Telecommunications, ICT 2016, Thessaloniki, Greece, May 16-18, 2016*, pages 1–6, 2016. 2, 40

[9] K. Mikhaylov, J. Petäjäjärvi, and T. Hänninen. Analysis of Capacity and Scalability of the LoRa Low Power Wide Area Network Technology. In *European Wireless 2016; 22th European Wireless Conference*, pages 1–6, May 2016. 2, 41

[10] Martin Bor, John Vidler, and Utz Roedig. LoRa for the Internet of Things. In *Proceedings of the 2016 International Conference on Embedded Wireless Systems and Networks*, EWSN '16, pages 361–366, USA, 2016. Junction Publishing. 2, 41

[11] 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*, 15(1), 10 2015. 2, 24, 41

[12] Moises Nunez Ochoa, Arturo Guizar, Mickael Maman, and Andrzej Duda. Evaluating LoRa Energy Efficiency for Adaptive Networks: From Star to Mesh Topologies. In *13th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, WiMob 2017, Rome, Italy, October 9-11, 2017*, pages 1–8, 2017. 2, 41

[13] L. Kleinrock and F. Tobagi. Packet Switching in Radio Channels: Part I–Carrier Sense Multiple-Access Modes and Their Throughput-Delay Characteristics. *IEEE Transactions on Communications*, 23(12), 1975. 3, 71

[14] Nokia. LTE Evolution for IoT Connectivity White Paper. https://onestore.nokia.com/asset/200178, 2015. 5, 7, 12

[15] The Internet Society. The Internet of Things: An Overview. https://www.internetsociety.org/wp-content/uploads/2017/08/ISOC-IoT-Overview-20151221-en.pdf, 2015. 6, 8

[16] Congduc Pham, Abdur Rahim, and Philippe Cousin. WAZIUP: A Low-Cost Infrastructure for Deploying IoT in Developing Countries. In *Proceedingsof the 8th International Conference on e-Infrastructure and e-Services for Developing Countries, AFRICOMM 2016*, volume 208 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 135–144. Springer, 2016. 7

[17] Corentin Dupont, Massimo Vecchio, Congduc Pham, Babacar Diop, Charlotte Dupont, and Sename Koffi. An Open IoT Platform to Promote Eco-Sustainable Innovation in Western Africa: Real Urban and Rural Testbeds. *Wireless Communications and Mobile Computing*, 2018, 2018. 7, 12

[18] Internet Architecture Board. RFC7452: Smart Object Architectural Considerations. https://www.rfc-editor.org/rfc/rfc7452.txt, 2015. 8, 9

[19] Internet Engineering Task Force. RFC6749: The OAuth 2.0 Authorization Framework. https://tools.ietf.org/html/rfc6749, 2012. 12

[20] U. Raza, P. Kulkarni, and M. Sooriyabandara. Low Power Wide Area Networks: An Overview. *IEEE Communications Surveys Tutorials*, 19(2):855–873, Secondquarter 2017. 12, 13, 15, 17

[21] P. A. Catherwood, D. Steele, M. Little, S. Mccomb, and J. Mclaughlin. A Community-Based IoT Personalized Wireless Healthcare Solution Trial. *IEEE Journal of Translational Engineering in Health and Medicine*, 6:1–13, 2018. 12

[22] H. M. Jawad, R. Nordin, S. K. Gharghan, A. M. Jawad, and M. Ismail. Energy-Efficient Wireless Sensor Networks for Precision Agriculture: A Review. *Sensors*, 17(1781):1–45, 2017. 12

[23] Vishal Sharma, Ilsun You, Giovanni Pau, Mario Collotta, Jae Lim, and Jeongnyeo Kim. LoRaWAN-Based Energy-Efficient Surveillance by Drones for Intelligent Transportation Systems. *Energies*, 11, 03 2018. 12

[24] W. Bakkali, M. Kieffer, M. Lalam, and T. Lestable. Kalman Filter-based Localization for Internet of Things LoRaWAN End Points. In *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 1–6, Oct 2017. 12

[25] Yosra Zguira, Hervé Rivano, and Aref Meddeb. IoB-DTN: A Lightweight DTN Protocol for Mobile IoT Applications to Smart Bike Sharing Systems. In *2018 Wireless Days, WD 2018, Dubai, United Arab Emirates, April 3-5, 2018*, pages 131–136. IEEE, April 2018. 12

[26] Yosra Zguira, Hervé Rivano, and Aref Meddeb. Internet of Bikes: A DTN Protocol with Data Aggregation for Urban Data Collection. *Sensors*, 18(9):2819, 2018. 12

[27] Y. Zguira, H. Rivano, and A. Meddeb. *Study and Development of Wireless Sensor Network Architecture Tolerant to Delays*. Thesis, Université de Lyon ; Université du Centre (Sousse, Tunisie), December 2018. 12

[28] G. Pasolini, C. Buratti, L. Feltrin, F. Zabini, C. De Castro, R. De Verdone, and O. De Andrisano. Smart City Pilot Projects Using LoRa and IEEE 802.15.4 Technologies. *Sensors*, 18(1118):1–17, 2018. 12

[29] M. de Castro Tomé, P. H. J. Nardelli, and H. Alves. Long-Range Low-Power Wireless Networks and Sampling Strategies in Electricity Metering. *IEEE Transactions on Industrial Electronics*, 66(2):1629–1637, Feb 2019. 12

[30] LoRa™ Alliance. A Technical Overview of LoRa and LoRaWAN. https://lora-alliance.org/sites/default/files/2018-04/what-is-lorawan.pdf. 13, 23

[31] Sigfox. Sigfox Technical Overview. https://www.disk91.com/wp-content/uploads/2017/05/4967675830228422064.pdf, 2017. 13, 14

[32] G. A. Akpakwu, B. J. Silva, G. P. Hancke, and A. M. Abu-Mahfouz. A Survey on 5G Networks for the Internet of Things: Communication Technologies and Challenges. *IEEE Access*, 6:3619–3647, 2018. 16, 19

[33] Min Chen, Yiming Miao, Yixue Hao, and Kai Hwang. Narrow Band Internet of Things. *IEEE Access*, PP:1–21, 2017. 16

[34] 3GPP TR 45.820 v13.1.0. Cellular System Support for Ultra Low Complexity and Low Throughput Internet of Things. http://www.3gpp.org/ftp/Specs/archive/45_series/45.820/45820-d10.zip, 2015. 17, 18, 19, 20

[35] Yihenew Dagne Beyene, Riku Jäntti, Olav Tirkkonen, Kalle Ruttik, Sassan Iraji, Anna Larmo, Tuomas Tirronen, and Johan Torsner. NB-IoT Technology Overview and Experience from Cloud-RAN Implementation. *IEEE Wirel. Commun.*, 24(3):26–32, 2017. 17

[36] Internet Engineering Task Force (IETF). Low-Power Wide Area Network (LPWAN) Overview. https://tools.ietf.org/html/rfc8376, 2018. RFC 8376. 18, 19

[37] Kais Mekki, Eddy Bajic, Frederic Chaxel, and Fernand Meyer. A Comparative Study of LPWAN Technologies for Large-Scale IoT Deployment. *ICT Express*, 5(1):1 – 7, 2019. 19

[38] B.S. Chaudhari and M. Zennaro. *LPWAN Technologies for IoT and M2M Applications*. Elsevier Science & Technology, 2020. 19

[39] A. J. Berni and W. Gregg. On the Utility of Chirp Modulation for Digital Signaling. *IEEE Trans. Commun.*, 21, 1971. 23

[40] Semtech Corporation. LoRa SX1276/77/78/79 Datasheet, Rev. 5. https://www.semtech.com/products/wireless-rf/lora-transceivers/sx1276, 2016. 25

[41] N. Sornin et al. LoRaWAN Backend Interfaces 1.0 Specification. https://lora-alliance.org/resource-hub/lorawanr-back-end-interfaces-v10, 2017. 36, 58

[42] D. Croce, M. Gucciardo, S. Mangione, G. Santaromita, and I. Tinnirello. Impact of LoRa Imperfect Orthogonality: Analysis of Link-Level Performance. *IEEE Communications Letters*, 22(4):796–799, April 2018. 39, 57

[43] Q. Lone, E. Dublé, F. Rousseau, Ingrid Moerman, Spilios Giannoulis, and A. Duda. WiSH-WalT: a Framework for Controllable and Reproducible LoRa Testbeds. In *2018 29th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pages 1–7, 2018. 41

[44] Andrzej Duda and Martin Heusse. Spatial Issues in Modeling LoRaWAN Capacity. In *Proc. ACM MSWiM '19*, November 25–29, 2019. xvi, 42, 90, 91

[45] T. Attia, M. Heusse, B. Tourancheau, and A. Duda. Experimental Characterization of LoRaWAN Link Quality. In *2019 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2019. 42

[46] C. Caillouet, M. Heusse, and F. Rousseau. Optimal SF Allocation in LoRaWAN Considering Physical Capture and Imperfect Orthogonality. In *2019 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2019. 42

[47] M. Heusse, T. Attia, C. Caillouet, F. Rousseau, and A. Duda. Capacity of a LoRaWAN Cell. In *IFIP Networking 2020, submitted*, 2020. 43

[48] M. Rizzi, P. Ferrari, A. Flammini, E. Sisinni, and M. Gidlund. Using LoRa for Industrial Wireless Networks. In *2017 IEEE 13th International Workshop on Factory Communication Systems (WFCS)*, pages 1–4, May 2017. 43

[49] B. Reynders, Q. Wang, P. Tuset-Peiro, X. Vilajosana, and S. Pollin. Improving Reliability and Scalability of LoRaWANs Through Lightweight Scheduling. *IEEE Internet of Things Journal*, 5(3):1830–1842, June 2018. 43

[50] Rajeev Piyare, Amy L. Murphy, Michele Magno, and Luca Benini. On-Demand LoRa: Asynchronous TDMA for Energy Efficient and Low Latency Communication in IoT. *Sensors*, 18(11), 2018. 44

[51] J. Haxhibeqiri, I. Moerman, and J. Hoebeke. Low Overhead Scheduling of LoRa Transmissions for Improved Scalability. *IEEE Internet of Things Journal*, 6(2):3097–3109, April 2019. 44

[52] D. Zorbas, K. Q. Abdelfadeel, V. Cionca, D. Pesch, and B. O'Flynn. Offline Scheduling Algorithms for Time-Slotted LoRa-based Bulk Data Transmission. In *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, pages 949–954, April 2019. 44

[53] NS-3 Consortium. NS-3 Network Simulator. https://www.nsnam.org, 2019. 47

[54] M. Knight. An Open-Source Implementation of the LoRa CSS PHY. https://github.com/BastilleResearch/gr-lora, 2016. Bastille Research. 55

[55] J. Blum. LoRa Modem with LimeSDRL. https://github.com/myriadrf/LoRa-SDR, 2016. 55

[56] Semtech. Semtech SX1272 Datasheet. https://www.semtech.com/products/wireless-rf/lora-transceivers/sx1272, 2019. 55

[57] Semtech Corporation. LoRa Gateway. http://www.semtech.com/apps/product.php?pn=SX1272. 56

[58] 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*, 15(1), 10 2015. 57

[59] He Wu, Sidharth Nabar, and Radha Poovendran. An Energy Framework for the Network Simulator 3 (NS-3). In *4th International ICST Conference on Simulation Tools and Techniques, SIMUTools '11, Barcelona, Spain, March 22 - 24, 2011*, pages 222–230, 2011. 58, 99

[60] Borja Martinez, Màrius Montón, Ignasi Vilajosana, and Joan Daniel Prades. The Power of Models: Modeling Power Consumption for IoT Devices. *IEEE Sensors Journal*, 15(10):5777–5789, 2015. 59

[61] Liviu-Octavian Varga, Gabriele Romaniello, Mališa Vučinić, M. Favre, A. Banciu, R. Guizzetti, C. Planat, Pascal Urard, Martin Heusse, Franck Rousseau, Olivier Alphand, Étienne Dublé, and Andrzej Duda. GreenNet: an Energy Harvesting IP-enabled Wireless Sensor Network. *IEEE Internet of Things Journal*, 2(5):412–426, 2015. 59

[62] ns-3 Project. Adding a New Module to NS-3. https://www.nsnam.org/docs/manual/html/new-modules.html, 2019. 59

[63] C. Lau and C. Leung. Capture Models for Model Packet Radio Networks. *IEEE Trans. Commun.*, 40:917–925, 1992. 65

[64] Brecht Reynders, Qing Wang, and Sofie Pollin. A LoRaWAN Module for Ns-3: Implementation and Evaluation. In *Proceedings of the 10th Workshop on Ns-3*, WNS3 '18, page 61–68, New York, NY, USA, 2018. Association for Computing Machinery. 68

[65] Floris Van den Abeele et al. NS-3 LoRa Module. Https://github.com/imec-idlab/ns-3-dev-git/tree/lorawan, 2017. 68

[66] Davide Magrin, Martina Capuzzo, Stefano Romagnolo, and Michele Luvisotto. NS-3 LoRa Module. https://github.com/signetlabdei/lorawan, 2018. 68

[67] John J. Metzner et al. *Wiley Encyclopedia of Telecommunications*. John Wiley and Sons, 1st edition, 2013. 72

[68] David J Wetherall Andrew S Tanenbaum. *Computer Networks*. Pearson, 5th edition, 2010. 74, 75, 76, 77

[69] Semtech Corporation. Semtech SX1272. http://www.semtech.com/apps/product.php?pn=SX1272, 2015. 79

[70] Semtech Corporation. Low Energy Consumption Design for SX1272/3/6/7/8 LoRa Modem. https://www.semtech.com/uploads/documents/LoraLowEnergyDesign_STD.pdf, 2013. 79

[71] Congduc Pham. Investigating and Experimenting CSMA Channel Access Mechanisms for LoRa IoT Networks. In *2018 IEEE Wireless Communications and Networking Conference, WCNC 2018, Barcelona, Spain, April 15-18, 2018*, pages 1–6. IEEE, 2018. 82

[72] Congduc Pham. Robust CSMA for Long-Range LoRa Transmissions with Image Sensing Devices. In *Proc. 2018 Wireless Days, Dubai, United Arab Emirates, April 3-5, 2018*, pages 116–122. IEEE, 2018. 82

[73] Timothy Claeys, Franck Rousseau, Bernard Tourancheau, and Andrzej Duda. Clock Drift Prediction for Fast Rejoin in 802.15.4e TSCH Networks. In *Proc. ICCCN*, pages 1–9, 2017. 89