POLITECNICO DI TORINO

Faculty of Computer Engineering

Master's of Science Degree in Software and Digital Systems

Master's of Science Degree Thesis

Improving Downlink Scalability in LoRaWAN

Supervisor

Paolo Giaccone

Candidate

Valentina Di Vincenzo

April 2020

Alla mia famiglia.
Radici ed ali.

Abstract

LoRaWAN is one of the most prominent LPWAN technology, offering captivating features for IoT applications including low power consumption, long-range and secure data transportation. Although LoRaWAN offers many advantages, one of its main limitations is the insufficient downlink capacity in large-scale scenarios. This makes reliable communication impractical as, in LoRaWAN, reliability is achieved through the implementation of a data retransmission scheme. Confirmed messages require the acknowledgment in form of downlink frame from the Network Server. Currently, however, the use of confirmed messages is strongly discouraged, as capable of saturating the ability of end-devices to efficiently send messages.

The contribution of this thesis is to provide a comprehensive analysis of the impact of downlinks on the network capacity and to propose solutions to enable the use of acknowledgments for low to medium downlink load.
The roots of the problem are identified and evaluated on a real traffic trace file: gateway transmissions are half-duplex (either sending or receiving), the downlink frame are transmitted exclusively sequentially and the duty-cycle limitation saturates the downlink frame sub-band, causing the more elevated packet loss.

To mitigate these negative effects, three solutions have been elaborated: a multi-gateway implementation, the downlink frames parallel sending and a balanced gateway selection algorithm. Two of them are tested thanks to the implementation of an event-driven simulator fed with captured real traces. From an initial 86% of frame loss with a single-gateway architecture, the proposed solutions provide an enhancement of 66% in a quad-gateway scenario under the maximum downlink load. Moreover, for low and medium downlink traffic loads, frame loss never exceeded 5%.

The proposed mechanisms build on a more adequate and flexible choice of the gateway for each downlink transmission by the Network Server. Thus, they are entirely compatible with existing LoRaWAN technologies and deployments.

The deployment of these solutions in real networks enables the realization of IoT LoRaWAN applications that require a more reliable communication and

the common practice of strongly discouraging the use of confirmed messages could be reconsidered.

Acknowledgement

I would like to thank my supervisor, Professor *Paolo Giaccone*, for his availability and professionalism. Furthermore, I would like to state my genuine gratitude to Professor *Martin Heusse*, my supervisor at the *Laboratoire d'Informatique de Grenoble*, for believing in me and for his guidance during the months spent at the *LIG* Laboratory. I would like also to mention the invaluable and fundamental support provided by Professor *Bernard Tourancheau*, as well as for his trust in offering me the opportunity to present this thesis in Shanghai at the *IEEE International Conference on Communications*. Finally, I express all my thankfulness to all the *LIG Drakkar* group researchers who inspired and challenged me from the very first day in Grenoble.

Contents

Chapter 1

Introduction

1.1 The Internet of Things

The increasing number of devices connected to the Internet is gradually building the future described by the Internet of Things (IoT) paradigm. Low power, long-range, low bit rate are three of the main characteristics of many Internet of Things applications. Indeed, in a classic IoT scenario, thousands of "things" are expected to send few bytes of information to a base station every time a specific event occurs (sometimes one transmission per hour or even days) [59]. It is possible to distinguish between two main IoT domain applications, consumer IoT and industrial IoT [61]. Applications belonging to the first domain are conceived to improve the consumer's life in terms of time and money management by increasing the interoperability between all the user's electronic devices. On the other hand, the focus of industrial IoT is to improve Business-to-Business (B2B) services exploiting Machine-to-Machine (M2M) interactions without requiring human intervention. Despite the different application domains, with different QoS and privacy requirements, the predisposition of a high level of coverage is a common need. This is necessary to asses the demand for reaching almost all the end-devices in the network, as happens in smart metering or smart health-care applications.

Moreover, the requirement for a long battery lifetime is important to enable that fraction of IoT applications that deploy battery-driven devices located in remote areas. The procedure of battery replacement could come with an unfeasible cost in terms of money or it can be impractical due to environmental

obstacles. While battery technology continues to evolve, devices are always smaller and their miniaturization goes to the detriment of the total energy available. The IoT communication process should asses this need by guaranteeing low power consumption.

The growing number of connected objects (around 30 billions expected in 2020 [46] and 75 billion by 2025) and the requirements of their communication, make most wireless solutions not completely adequate since they have been built for different devices with different needs.

Up to now, one of the main strategies to supply connectivity to things has been provided by short-range technologies such as Bluetooth Low Energy (BLE) and IEEE 802.15.4-based systems (e.g ZigBee). This type of IoT connectivity is characterized by multi-hop mesh networks and a high data rate to the detriment of the coverage range. The significant control-traffic necessary to the upkeep of mesh networks and the uneven, uncertain energy consumption that multi-hop routing implies, are two of the limits shown by these technologies. Nevertheless, the major obstacle is their limited coverage, especially in IoT scenarios that require urban-wide coverage (*e.g* smart-city applications)[44].

On the other hand, the current cellular networks could allow long-range communication [62], but, being conceived to support communications between humans and less between M2M, their design struggle to serve the massive amount of clients that the IoT context will present and scalability problems will soon arise. The potentially huge number of IoT devices asking for connectivity through a single Base Station (BS) would raise new issues related to the signaling and control traffic [52], which may become the bottleneck of the system. All these aspects make the fourth-generation (4G) broadband mobile networks unfit to support the envisioned IoT scenarios. Indeed, the 3rd Generation Partnership Project (3GPP) system standards are heading towards embedding M2M communications in the 5G systems and to support *massive Machine Type Communication* (mMTC).

Low Power Wide Area (LPWA) solutions arise to fill the gap left by these technologies, meeting the application requirements in areas where the traditional cellular M2M systems have not been optimized [48]. Moreover, they provide a much lower cost of deployment and initial investments, inducing

	Technology	Bands
Short range	IEEE 802.15.4, Bluetooth	Unlicensed
	IEEE 802.11, IEEE 802.15.6	Unlicensed
Long Range	3GPP 4G, 5G	Licensed
	LPWANs	Unlicensed

Table 1.1: IoT connectivity classification.

many Mobile Network Operators (MNOs), such as Orange, Bouygues Telecom, KPN, to integrate LoRaWAN infrastructure as a complement to their current networks deployments [33].

Long-Range Wide Area Network (LoRaWAN) is one of the most prominent LPWAN technology, alongside with SigFOX and Weightless. As an LPWAN solution, LoRaWAN is addressed to fulfill the need for long-range communication between thousands of low power devices (possibly battery-driven).

1.2 LPWAN for the IoT

LPWAN solutions arise to provide connectivity between low power connected objects in wide area networks [40], deployed for different IoT applications such as smart homes, cities [29] and agriculture [19]. The main focus of LPWAN technologies is well expressed by the term itself that stands for *high reach, low cost, low power* Wide Area Networks and was introduced to the market by Machina Research [53]. LPWANs are the optimal solution IoT applications requiring low-cost nodes with long battery lifetime (even up to a decade from a single AA battery) and a small amount of data exchanged sporadically, as shown in Figure 1.1.

The LPWAN landscape includes several proprietary solutions that operate in the unlicensed spectrum and its growth it has been forecast to reach three billions LPWA M2M connections by 2023, according to Machina Research.

The focus of this work is on LoRaWAN, one of the most active LPWAN protocol in terms of deployment and development.

In LoRaWAN, the end-devices can reach different base-stations (gateways) at a distance of kilometers through a single wireless hop. Each gateway then

forwards the data to its specific server on the internet. Despite LoRaWAN offers many advantages, one of the main drawbacks is the insufficient downlink capacity in large-scale network scenarios: sending data back to the sensors/devices is severely limited [10], making the deployment of ultra-reliable applications unfeasible [20].

The contribution of this work is to provide a clear, deep and comprehensive explanation of the downlink traffic problem and to propose solutions aimed at supplying LoRaWAN with a stronger acknowledgment mechanism even in scenarios of heavy downlink load.

	LoRaWAN	**Sigfox**	**NB-IoT**	**LTE Cat-M1**
Coverage (MCL)	157 dB	162 dB	164 dB	155 dB
Technology	Proprietary	Proprietary	Open LTE	Open LTE
Spectrum	Unlicensed	Unlicensed	Licensed (LTE/any)	Licensed (LTE/any)
Duty cycle limit	Yes	Yes	No	No
Output power restrictions	Yes (14 dBm = 25 mW)	Yes (14 dBm = 25 mW)	No (23 dBm = 200 mW)	No (23 dBm = 200 mW)
Downlink data rate	0.3-50 kbps	< 1 kbps	0.5-27.2 kbps	< 300 kbps
Uplink data rate	0.3-50 kbps	< 1 kbps	0.3-32.25 kbps	< 375 kbps
Battery life / Current consumption	8+ years <2 uA	10+ years <2 uA	10+ years <3 uA	10+ years <8 uA
Module cost	<$ 10	<$ 10	$ 10 (2019); $ 3 (2020)	<$ 25 (2019)
Security	Medium (AES-128)	Low (AES-128)	Very high (LTE Security)	Very high (LTE Security)

Figure 1.1: Key parameters and characteristics of most popular LPWA technologies provided by [4]

1.3 Improving Downlink Traffic Scalability

In LoRaWAN, reliability is achieved through the acknowledgment of frames in the downlink. Nevertheless, currently, the deployment of confirmed messages in LoRaWAN is not recommended. The reason behind this deprecation is the negative impact that the downlink traffic would bring to the overall network. Indeed, all the traffic is negatively affected by the presence of confirmed messages, including unconfirmed ones [24].

Broker et al. in [2] depict how LoRaWAN has a high potential of contribution in the area of 5G mMTC applications (covering around the 10% of the

5G mMTC target 1,000,000 devices per square kilometer), but just in terms of uplinks. The performance evaluations show that, due to the unlicensed band regulations, the downlink capability limits LoRaWAN application to use cases without or with low QoS requirements.

The first step towards higher reliability in LoRaWAN is to understand the origin of such a problem, followed by the elaboration of specific solutions to solve it. In order to classify the various causes of the problem in terms of packet loss and to quantify the improvements that the solutions could bring, a simulator has been developed. Three main sources of packet loss due to the transmission of downlinks have been identified: the gateway-half duplex mode (exclusively alternative sending and receiving) causes up to 9% of packet loss; the downlink sequential sending of gateway's concentrator- by which only one downlink may be scheduled at a given time - produces up to 3% of packet loss due to ack collision with other downlinks already scheduled in the concentrator; and the duty-cycle limitation induces up to 76% of packet loss due to the saturation of the downlink sub-band: each time a downlink is sent, a *time off* related to the used sub-band is calculated.

In order to face these problems, three solutions are evaluated and tested thanks to the event-driven simulator. The deployment of a multi-gateway network architecture smooths the half-duplex problem by halving the packets loss due to it and the duty-cycle saturation. By introducing a balanced gateway selection algorithm, the duty-cycle limitation problem is further decreased by 25%.

Just by adding three gateways to a single-gateway implementation, the proposed solutions bring an improvement of 66% in terms of packets loss, losing *just* the 20% of the packets in a scenario with an outraged downlink load (100% of confirmed packets).

The work exposed in this thesis has produced a conference paper called "*Improving Downlink Scalability in LoRaWAN*" that has been accepted to the 53rd *IEEE International Conference on Communications* (ICC) hosted in Shanghai.

1.4 Content Structure

After this introduction,

Chapter 2 aims at providing all the information about LoRa technology necessary to understand the motivations of this work, the problem statement, and the proposed solutions: the first section provides an overview of the LoRa stack and the reasons behind the growth of Low Power Wide Area Network technologies, such as LoRa; in the second section, the physical and MAC layer of LoRa are presented. Emphasis is given to the most significant details related to the generation of downlink traffic.

Chapter 3 is dedicated to the problem statement. The motivations behind this work are explained, the problem is analyzed and its origins are identified in the first section. Each subsection inspects a different component: the duty-cycle saturation problem is faced, providing a clear vision of the consequences caused by the ISM band regulations; the half-duplex mode of the gateway is identified as the only cause of unconfirmed uplink traffic loss due to downlinks generation and the distinction between *free* and *busy* gateway is given; the downlinks sequential scheduling prevents the transmission of multiple acks at the same time, exacerbating confirmed packet loss. Finally, an explanatory example is given with the purpose of collecting all the problems and showing them in action.

Chapter 4 exposes three solutions to the problems shown in Chapter 3. The solutions are explained with a theoretical approach and their drawbacks are highlighted. The multi-gateway configuration is mainly intended to alleviate the half-duplex problem; the balanced gateways selection algorithm would provide a better spread of the downlink load, pushing back the outset of the saturation of the gateways and last, the parallel sending would allow downlinks combination. As before, an explanatory example is given, exploiting the same architecture of the first example and showing how the described solutions would decrease packets loss. Ultimately, the related work is presented in order to perceive the contribution of this work.

Chapter 5 addresses the evaluation of different experimental approaches and the explanation of the chosen one. To validate the proposed solutions and analyze the scalability issue due to the downlink traffic, an event-driven simulator that receives as input real traffic traces has been developed. The

first section is dedicated to explaining the real traffic traces collection process, whereas the second section explains the simulator operating principles. Three main tasks are simulated: downlink traffic generation, packets analysis, and downlinks scheduling.

In Chapter 6 the simulation results are presented and discussed. In the first section, the physical data collection allows the inspection of the downlink traffic problem in a single-gateway architecture and provides meaningful insights about the causes of the packets loss. The results relative to the multi-gateway implementation are shown in Section 6.2, where it is possible to appreciate how the packet loss varies with the number of deployed gateways under different percentages of downlink load. To understand the weaknesses of the SNR-based selection algorithm, the downlink load per gateway is shown. In Section 6.3 an additional step is taken, and the balanced gateway selection algorithm is tested and compared with the SNR one.

Finally, Chapter 7 presents a summary of the work and a discussion of the obtained results. At last, future work possibilities are explored.

Chapter 2

LoRa Overview

LoRaWAN is one of the most prominent LPWAN technology operating in the Industrial, Scientific and Medical (ISM) band, alongside SigFOX and Weightless, among others. Differently than other competitors, LoRaWAN offers the possibility to the LoRa end-devices to receive downlink frames. SigFOX constraints the downlink communication by imposing a limit of two confirmed data messages per day [7], and the downlink communication must always be initiated by the end-device, severely limiting the range of usage and discouraging the transmission of critical real-time data. On the other hand, LoRaWAN shows network capacity problems each time the downlink traffic load increase. The aim of this chapter is to provide a comprehensive vision of the presented technology in order to fully understand the problems that arise in LoRaWAN downlink communication.

2.1 LoRa Stack

LoRa can refer to two different layers:

- **LoRa** is the physical layer (LoRaPHY). At the hardware level, it defines the electrical specifications for data transfer. It is a Semtech proprietary technology and, as such, not fully open. It exploits unlicensed Industrial, Scientific and Medical (ISM) frequency bands and uses a Chirp Spreading Spectrum (CSS) radio modulation technique [63].
- **LoRaWAN** is the protocol stack (at the Media Access Control (MAC)

and network layers) which exploits the LoRa physical layer. It is an open standard developed by the LoRa Alliance™[1] [35].

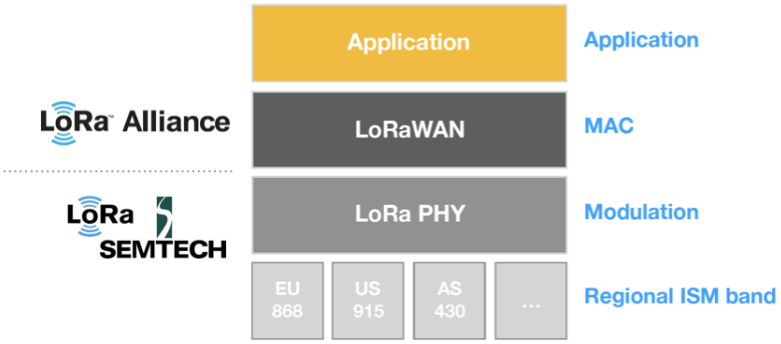


Figure 2.1: LoRa technology stack.

2.2 LoRa Modulation

LoRa is a wireless networking standard based on the Chirp Spread Spectrum (CSS) modulation. Chirp (or sweep signal) stands for "Compressed High-Intensity Radar Pulse" and it is a signal which frequency either increases or decreases with time. If the frequency changes from lowest to highest, it is call up-chirp (the derivative of the frequency variation is positive) and if the frequency changes from highest to lowest, it is called a down-chirp. Following, the example of a linear up-chirp waveform where the frequency increases continuously and linearly over time:

To explain the Lora modulation, three main parameters have to be taken into account:

- **Spreading Factor (SF)**. It defines the chirp duration, with shorter Time on Air corresponding to a smaller value. Lora operates with SF that spans from 7 to 12. A LoRa symbol is composed of 2^{SF} chirps and, as a consequence, a symbol can effectively encode SF bits of information. In Figure 2.3 it is possible to observe how a higher spreading factor corresponds a longer chirp [49]: SF8 takes exactly twice the time of SF7 and SF9 takes exactly twice the time of SF8.

[1] `https://lora-alliance.org`

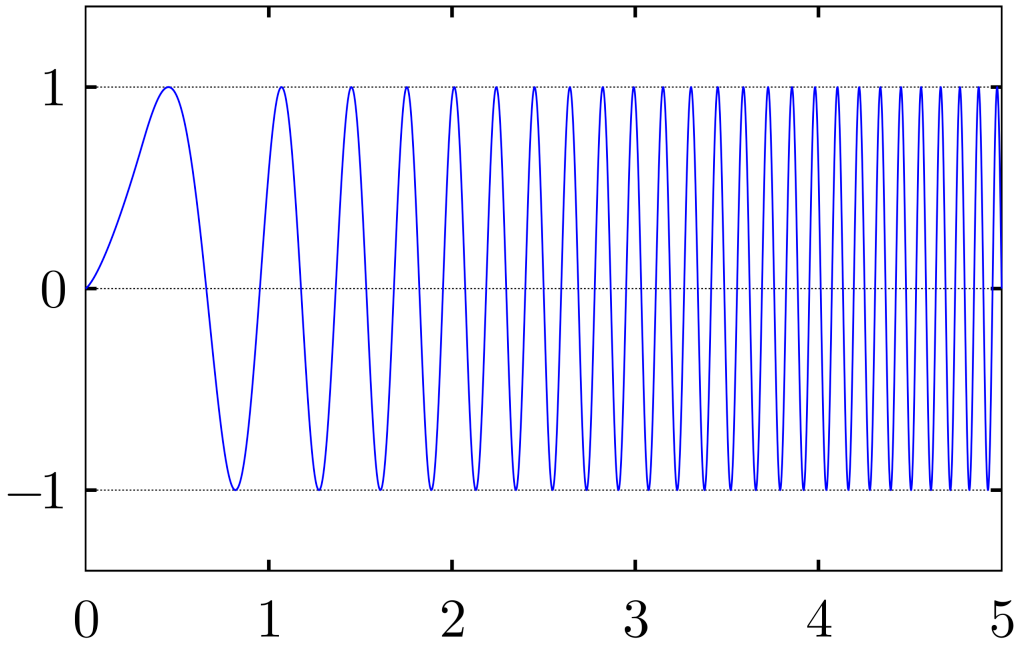


Figure 2.2: Plot of the Linear Chirp with t in [0,5]

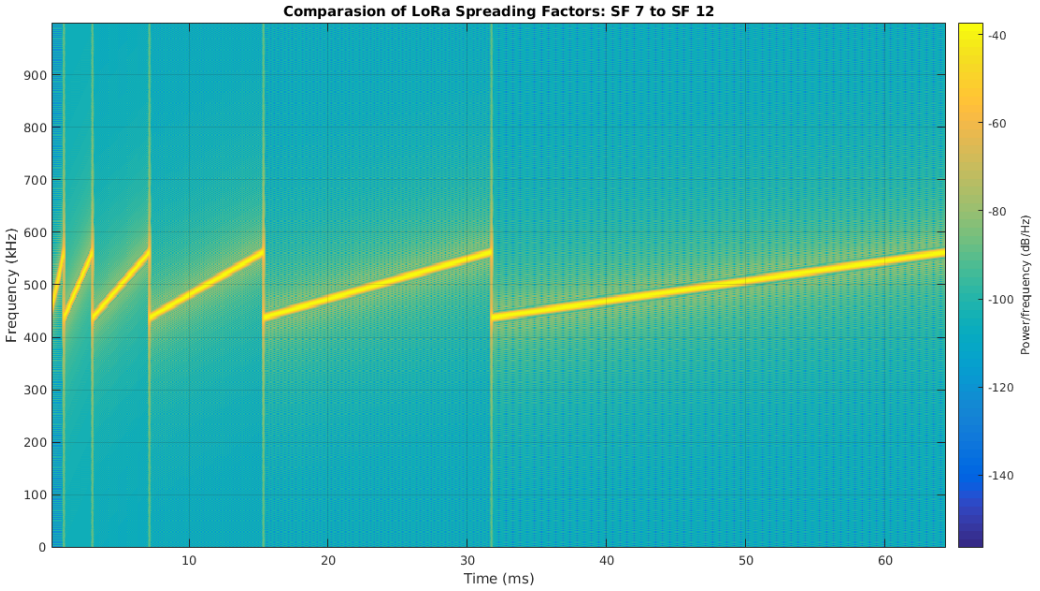


Figure 2.3: Spectrogram of LoRa spreading factors [28].

It is important to highlight that for every single increase of SF (and hence additional symbol bit), the number of chirps used to modulate a symbol is doubled. This means that LoRa can exhibit a particularly slow modulation (possibly with Time on Air longer than a minute). This drawback is balanced by the fact that it gives the robustness property that enables the demodulation of LoRa packets even beneath the level of the noise floor. Notice that an error-correcting code introduces some redundancy at the bit encoding level.

- **Bandwidth (BW)**. It is the range of transmission frequencies. LoRa uses three bandwidths: 125kHz, 250kHz, and 500kHz. The frequency bandwidth of the chirp is equivalent to the spectral bandwidth of the signal since CSS uses the complete bandwidth when transmitting.

- **Carrier Frequency (CF)**. This is a regional parameter. In Europe, the frequency spans the 863 - 870 MHz (EU868), as specified in [14].

In the modulation phase, the symbol of SF bits has to be translated into multiple chirps, each of them spanning 2^{SF} samples at the specified BW sampling rate. The frequency spreading is achieved by coding the information into the time offset applied to the base chirp waveform. Therefore, there are 2^{SF} different and unique shifts (or time offsets), which allows conveying SF bits. Moreover, it is possible to distinguish between N different symbols thanks to the definition of N orthogonal chirps: each symbol shows a unique phase trajectory in a specif moment. The following plot shows how different symbols are modulated.

The LoRa demodulation used to detect the transmitted symbols consists of four main steps:

1. Multiply the waveform by the conjugate of the raw chirp used in the transmitter (up or down) to obtain regions of constant frequency;

2. Compute the FFT for each of the obtained regions;

3. Align the received chips to guarantee a good synchronization of the chirps both in time and frequency [12];

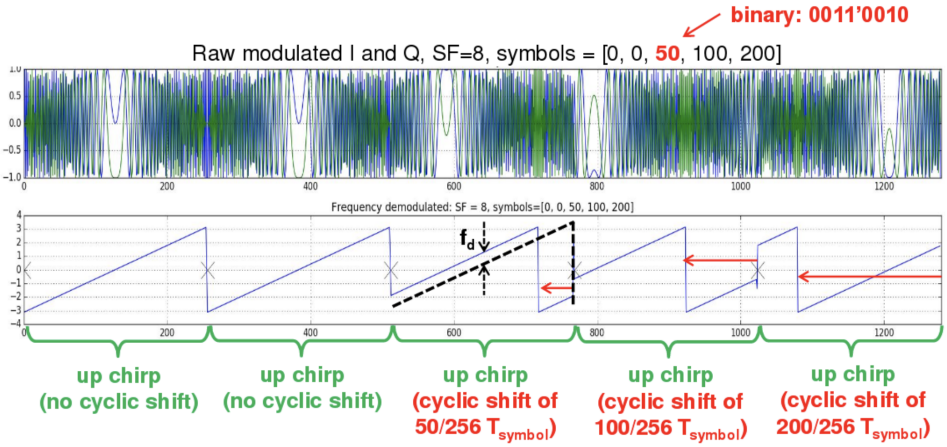


Figure 2.4: LoRa modulation symbols example, from [41].

4. Observe the peak to identify the symbol's value, even in the presence of important noise.

In the following plot, we can observe the demodulated symbol as the peak index in the FFT of each region.

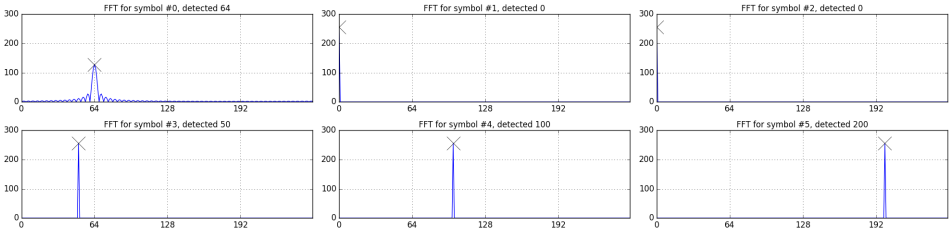


Figure 2.5: Demodulated symbols (by FFT), from [41].

Moreover, the SF value is in strict correlation with the Data Rate (DR), defining the number of bits effectively transmitted by each symbol and the chirp duration. Increasing the SF decreases the DR since the same bytes have to be transmitted in a longer time. As an indirect consequence, the gateway that receives a transmission with a lower SF is more sensitive to noise: LoRa trades DR for sensitivity within a fixed channel BW, as shown in Table 2.3 for the 125 kHz bandwidth modulations. A higher spreading factor increases the receiver sensitivity but decreases the data rate since it increases the chirp period.

The Data Rate and the end-point output power (TXPower) in the EU863-870 band are encoded as shown in Table 2.1 and 2.2. This encoding is used

DataRate	Configuration	Indicative physical bit rate [bit/s]
0	SF12 / 125 kHz	250
1	SF11 / 125 kHz	440
2	SF10 / 125 kHz	980
3	SF9 / 125 kHz	1760
4	SF8 / 125 kHz	3125
5	SF7 / 125 kHz	5470
6	SF7 / 250 kHz	11000

Table 2.1: Data Rate encoding in the EU863-870 band, from [14].

TXPower	Configuration (EIRP)
0	MaxEIRP
1	MaxEIRP - 2dB
2	MaxEIRP - 4dB
3	MaxEIRP - 6dB
4	MaxEIRP - 8dB
5	MaxEIRP - 10dB
6	MaxEIRP - 12dB
7	MaxEIRP - 14dB

Table 2.2: TX power encoding in the EU863-870 band, from [14].

in all the MAC command fields related to this type of information. It is important to highlight how the maximum Signal to Noise Ration (SNR) value varies in function of the SF and the TXPower. In particular, increasing the transmission power results in better SNR [13].

The technology robustness mixed up with the ability to operate in low power mode makes LoRa a perfect candidate to be exploited for connecting the Internet of Things. Also, LoRa provides more capacity in the system: the receiver can detect multiple simultaneous transmission from several nodes by

Table 2.3: LoRa transceiver (Semtech SX1276) parameters.

SF	**7**	**8**	**9**	**10**	**11**	**12**
Sensitivity [dBm]	-123	-126	-129	-132	-133	-136
DR [kbit/s]	3.4-5.5	2.0-3.1	1.1-1.8	0.6-1.0	0.3-0.5	0.2-0.3
min SNR [dB]	-7.5	-10	-12.5	-15	-17.5	-20

exploiting the orthogonality of signals at different spreading factors [50].

2.2.1 Time On Air

The Time On Air defines the transmission time of a packet [57]. The symbol duration is the time spent to send 2^{SF} chips at the chip-rate. In the LoRa modulation, the latter is equal to the bandwidth (a chip per-second-per-Hertz). For example, a LoRa bandwidth of 125 kHz corresponds to a chip rate of 125 kcps.

$$T_{sym} = \frac{2^{SF}}{BW} \tag{2.1}$$

Before the transmission of the payload, a preamble made of raw chirps has to be sent, in order to estimate the offset between the internal times of the transmitter and the receiver and perform their synchronization. As a consequence, the packet duration is given by the sum of the preamble and payload durations.

The first component is given by:

$$T_{preamble} = (n_{preamble} + 4.25)T_{sym} \tag{2.2}$$

Where $n_{preamble}$ is the programmed number of symbols in the preamble (usually 8).

The number of symbols contained in the payload and in the header is express as:

$$PayloadSymNb = 8 + max(ceil\Big(\frac{8PL - 4SF + 28 + 26 - 20H}{4(SF - 2DE)}\Big)(CR + 4), 0) \tag{2.3}$$

With the following dependencies:

- PL, bytes payload
- SF, spreading factor
- H, 0 when the header is present, 1 otherwise (the header is always presents except for the beacons of class C devices).
- DE, 1 when the low data rate optimization is enable (SF>10), 0 otherwise

- CR, code rate from 1 to 4

The payload duration is given by the number of symbols and the symbol period:

$$T_{payload} = payloadSymNb * T_{sym} \tag{2.4}$$

Finally, the Time On Air can be calculated as:

$$T_{packet} = T_{preable} + T_{payload} \tag{2.5}$$

Hence, size, spreading factor, bandwidth, and code rate are the main factors to influence the duration of a packet [31], as shown in Figure 2.6

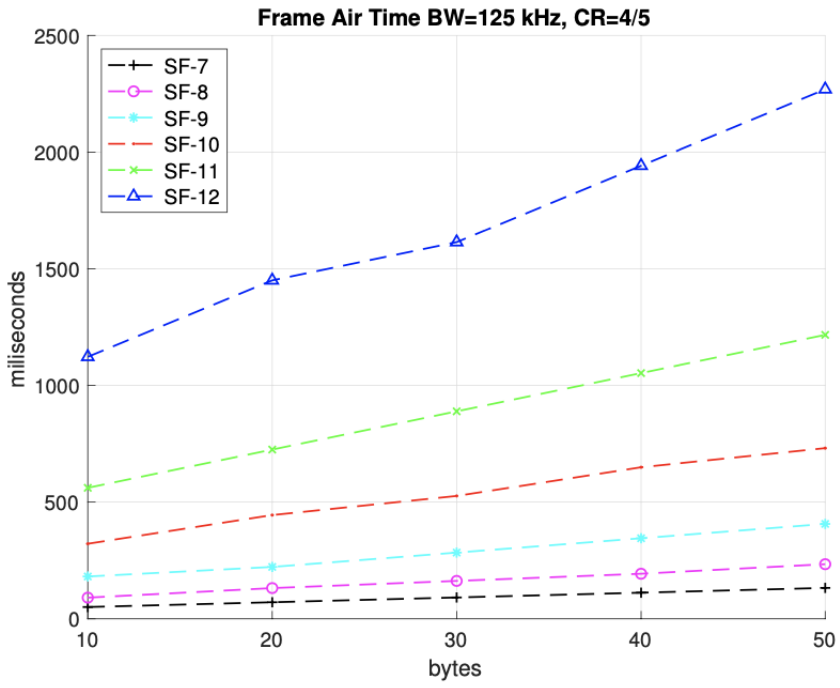


Figure 2.6: LoRa frame air time comparison for CR = 4/5 BW = 125 kHz, from [6].

2.3 LoRaWAN

LoRaWAN presents a star-of-stars topology, designed to give wide area coverage and to ensure the connectivity also to nodes that are deployed in very

harsh environments [51].

2.3.1 Network Architecture

The network architecture presents three main components:

- **End-device.** The devices are at the end of the network architecture. Each end device is registered with its specific application that collects data from all the registered end-devices and possibly sends new commands to them. In order to send data to the applications, the device emits LoRa packets towards a server each time a specific event triggers the transmission.

- **Gateways.** Gateways are receiving LoRa packets from all the LoRa motes sending traffic in the area. Each gateway is registered to a server to which it transparently forwards through IP connection all the traffic received.

  This architecture allows a single-hop link between the end-device and one or many gateways.

  In addition, if the concentrator only has a buffer of one downlink message, the gateway needs to build a Just In Time (JIT) queue in order to store and order incoming downlink packets, so that they can all be programmed in the concentrator at the proper time.

- **Network Server.** The server is where all the core logic is stored. In order to let the end-devices be as simple as possible, the server is in charge of all the decisions and parameter adaptation. Several gateways can be registered to the same server, that could receive the same packet more than once. In this context, the server needs to perform a de-duplication operation by selecting just one packet. The chosen one is then forwarded to the application.

2.3.2 End-device Classification

In LoRaWAN Network it is possible to identify two type of packets:

- **Uplink**: packet sent by the end-device to the Network Server.

- **Downlink**: packet sent in the opposite direction.

In order to allow the end devices to receive downlinks, different strategies are used based on energy consumption.

As a consequence, it is possible to distinguish three different end-devices classes:

- **Class A - All end-devices.** All devices are booted in this mode. It allows the lowest energy consumption possible by making the end device listen to downlink traffic in two fixed-in-time receive windows. Every time an uplink is sent, a first receive window (RX1) is opened by the end-device after a `RECEIVE_DELAY1`seconds (default value 1s) from the uplink modulation. The end-device listening in RX1 is expecting a packet on the same frequency of the uplink and with a data rate (DR) that is a function of the uplink DR. If a preamble is detected, the radio receiver stays active until the downlink frame is demodulated. If, and only if, no downlink is received in RX1, the end-device opens a second receive window (RX2) after `RECEIVE_DELAY2` seconds (default value `RECEIVE_DELAY1`+1s) from the uplink modulation. The default values for RX2 are 869.5MHz/DR0 (SF12, 125 kHz).

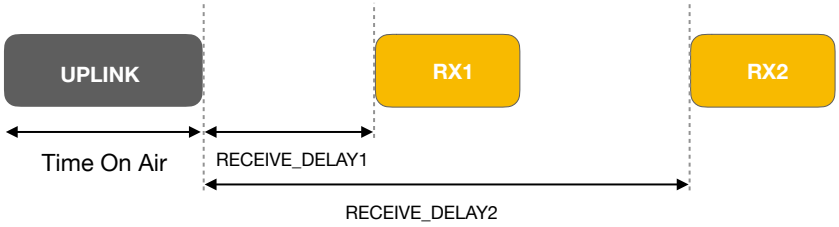


Figure 2.7: Class A end-devices receive slot timing.

- **Class B - Beacons.** A Class A end-device application can decide to switch to Class B. The purpose of Class B is to have an end-device available for the reception on a predictable time, in addition to the receive windows that follow the random uplinks transmission. The additional receive windows are opened thanks to the synchronization allowed by beacons sent on a regular basis from the gateway to all the end-devices in the network. The end-device must open reception slots (called *ping*

slot) at a precise instant relative to the infrastructure beacon. This allows the server to know when the end-device is listening.

- **Class C - Continuous listening.** This class allows devices to constantly be in receive mode, except when an uplink has to be sent. Obviously, these devices show the maximum power consumption.

2.3.3 MAC Message Format

When the Network Server receives a message, it has to analyze it in order to appropriately select the following steps to be taken in regard to the given downlink. Each field of the MAC Message gives important information about the end device state, and it is used to guarantee security and to request the server to perform some parameter adaptation. To evaluate the solutions proposed by this work it has been necessary to develop a simulator where the decision-making process and the behavior of a Network Server are accurately reproduced since it has to be able to correctly produce a response or changes (when needed) that can potentially impact the whole network (as lowering DR, sending an acknowledge..). At the same time, we need to be able to manipulate the packets and make them perform special requests to the Network Server. Indeed, the study of the message format has been of crucial importance in the making of the simulator.

The physical payload is composed of three main components: the message header (MHDR), the MAC Payload and a Message Integrity Code (MIC), as shown in Figure 2.8

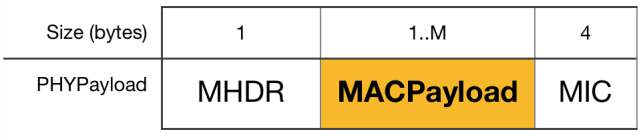


Figure 2.8: LoRaWAN Physical Payload. In yellow the field of interest.

- **MHDR** (1 byte)

  MType (3 bits): The first three bits of the message header are dedicated to the identification of the message type, encoding the information based on Table 2.4.

The *Join Request* and *Join Accept* encodings are used in the Over-The-Air-Activation (OTAA). A *Confirmed-data message* type requires the acknowledge of reception by the receiver, while an *Unconfirmed-data message* does not. Property messages are used for message format different from the standard, and thus can be used only between devices that have a common understanding of the proprietary extension.

MType	**Description**
000	Join Request
001	Join Accept
010	Unconfirmed Data Up
011	Unconfirmed Data Down
100	Confirmed Data Up
101	Confirmed Data Down
110	RFU
111	Proprietary

Table 2.4: MAC Commands type encoding, from [35]

RFU (3 bits): Reserved for future usage.

Major (2 bits): The frame is encoded according to this field, where the major version of the frame format of the LoRaWAN layer is specified.

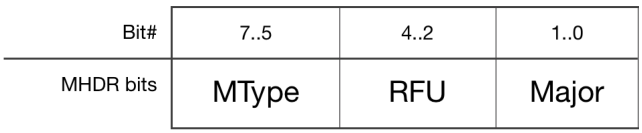


Figure 2.9: Message header format

- **MACPayload** (1..M bytes): M is the maximum size of the MAC Payload and it is region-specific. It varies with the DR used, as shown in Table 2.5 for EU863-870, where N is the maximum length for frame payload.

  The MAC payload is also called "data-frame". It is composed of a Frame Header (FHDR), and two optional fields: FPort and FRMPayload.

  – **FHDR** (7 up to 23 bytes): Frame header. See below.

Data Rata	M	N
0	59	51
1	59	51
2	59	51
3	123	115
4	230	222
5	230	222
6	230	222
7	230	222

Table 2.5: Payload length based on DR, from [35]

– **FPort** (0 or 1 byte): If 0 means that the FRMPayload contains just MAC Commands. Port values 1..223 (0x01..0xDF) are application specific. FPort values 224..255 (0xE0..0xFF) are reserved for future standardized application extensions.

– **FRMPayload** (0 up to N bytes): It has a maximum value of N (region specific, see table 2.5). N should be:

$$N <= M - FHDRbytes - 1 \tag{2.6}$$

where M os the maximum number of bytes for the MAC Payload size.

Encryption: FRMPayload must be encrypted before the Message Integrity Code (MIC) is calculated. The encryption scheme uses AES key length of 128 bits. As a default, the encryption/decryption is done by the LoRaWAN layer for all FPort. Based on the value of the FPort the key (K) used changes, as shown in Table 2.6.

FPort	K
0	NwkSKey
1..255	AppSKey

Table 2.6: K variation on FPort value, from [35]

The Network Session Key (NwkSKey) is shared with the network and it is used to allow secure communication between the end device and the Network Server. The AppSKey is kept private and used solely to secure

the message payload. These session keys are unique for each end device and are used for the duration of the session (generating a new pair at each activation with OTAA devices, or just one time with ABP devices. See 2.3.5.).

Size (bytes)	7..23	0..1	0..N
MACPayload	**FHDR**	FPort	FRMPayload

Figure 2.10: MAC Payload format. In yellow the field of interest.

- **MIC** (4 bytes) The message integrity code, similarly to a checksum, is calculated over all the fields in the message by exploiting the Network Session Key exchanged at the beginning of the communication. To be valid, the receive MIC signature must match the one obtained by calculating the MIC using the specific Network Session Key stored in the receiver key database and referring to the given device.

$$msg = MHDR|FHDR|FPort|FRMPayload \tag{2.7}$$

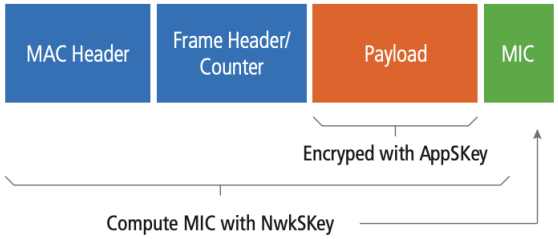


Figure 2.11: Structure of a LoRaWAN packet and its protection, from [26].

- **Frame Header (FHDR)** (7..23 bytes): It contains, as shown in Figure 2.12, a short device address (32 bit), a Frame Control octet, a 2-octets Frame Counter and an optional (up to 15 bytes) Frame Option field, used to transport MAC command.

  – **DevAddr** (4 bytes): Non-unique short device address of 32 bits. In the Network Server, it is mapped to a globally unique end-device identifier DevEUI (in IEEE EUI64 address space) and AppEUI (in IEEE

Size (bytes)	4	1	2	0..15
FHDR	DevAddr	**FCtrl**	FCnt	FOpts

Figure 2.12: Frame header format. In yellow the field of interest.

EUI64 address space) to identify the application provider. To understand how this mapping is performed, see 2.3.5.

– **FCtrl** (1 byte): see below.

– **Frame Counter (FCnt)** (2 bytes): Despite the security given by the validity check from the MIC and the message encryption, it is still possible to perform an attack by transmitting messages multiple times with the intent to saturate and exhaust the Network Server. To prevent these so-called *replay attacks*, the FCnt field of the frame header is exploited. It is possible to detect and block such attacks by registering the number of messages sent by a device. To fulfill this task, there are two frame counters for each device: FCntUp (CU), incremented by the end-device each time an uplink frame is sent and FCntDown (CD), incremented by the Network Server each time a frame to that end device is sent (not for retransmission). At the activation, these values are reset in both the end-device and the Network Server and they both will keep track of the two counters: the first will indicate the number of messages sent by the end-device, the second the ones sent by the server. For security purposes, the Network Server compares the FCnt value received in the uplink frame with the corresponding one it has stored for that particular device. If the new value is lower than the old one the message is ignored.

– **FOpts** (0 up to 15 bytes): The real size is specified by the field FOptsLen in the FCtrl. If the FOptsLen is different from zero, it means that in FOpts some MAC commands are present. If MAC commands are present in the FOpts field, they will not be present inf FRMPayload on the MACpayload and vice-versa. In this context, the port 0 in the FPort field can not be used, since the value 0 states that the FRMPayload contains just MAC commands.

- **Frame Control (FCtrl)** (1 byte): The most relevant fields for the current work are the ones related to the generation and the control of downlink traffic, as shown in Figure 2.14.

Bit#	7	6	5	4	[3..0]
FCtrl bits	ADR	ADRACKReq	ACK	FPending/RFU	FOptsLen

Figure 2.13: Frame Control bits.

  – **ADR** and **ADRACKReq** [1 bit each]: fields for Adaptive Data Rate. LoRaWAN allows end-devices to adapt the data rate in order to increase the battery life and maximize the network capacity. This feature is exploited by static devices, while mobile devices do not use it because the fast changes in the radio environment while moving would make data rate management not practical. When the ADR bit is set (by the end-device or the Network Server on demand), the feature is enabled and the network can control the data rate through the correspondent MAC command *LinkADRReq* and *LinkADRAns* (later explained). If the end-device uses a data rate which is greater than its default one, it needs to validate that the uplinks are still received by the Network Server. In order to do that, three parameters are exploited:

    – `ADR_ACK_COUNT`, incremented each time an uplink message is sent (not for retransmission of the same message).

    – `ADR_ACK_LIMIT`, fixed maximum number of uplinks sent without any downlink received. After that, the ADRACKReq bit has to be set.

    – `ADR_ACK_DELAY`, fixed number of uplinks sent while waiting for a downlink after having set the ADRACKReq bit.

  When

$$ADR\_ACK\_COUNT >= ADR\_ACK\_LIMIT \tag{2.8}$$

  the end-device will set the ADRACKReq bit that informs the network server to sent a downlink message within the next `ADR_ACK_DELAY`

uplinks. If a downlink is received, then ADR_ACK_COUNT is reset. Otherwise, the end-device may try to regain connectivity by switching to the next lower the data rate in order to have a longer radio range.

– **ACK** (1 bit). A confirmed message is a message that requires an acknowledgment from the receiver. If the confirmed message is sent by the end-device (confirmed uplink), the network must send, in the following receive windows, a downlink frame with the ACK bit set. If the confirmed message is sent by the Network Server (confirmed downlink), the end-device can send the acknowledge at its own discretion. Usually, in order to have the simplest end-devices, an explicit acknowledge data message is sent immediately after the confirmed downlink reception. Otherwise, the ACK can be piggybacked in the following uplink transmission.

If an acknowledge frame is not received after a number of retransmissions of the same confirmed message, devices can behave differently. End-devices can try to regain connectivity by lowering the data rate and the try to send again the message or discard it. On the other hand, the Network Server can consider that particular end-device as unreachable until it receives some messages from it. If the connectivity is regained, the Network Server will decide whether to send the message again or forfeit it and move on.

It is worth noting that acknowledges are sent only in response to the latest message received and are never retransmitted.

– **FCnt** (2 bytes): End-device frame counter incremented each time an new uplink is sent. Notice that retransmissions do not trigger such increment.

– **FPending/RFU** (1 bit): When the FPending bit is set, it means that the Network Server has some data pending to be sent for the end device and needs it to “wake up”. Thus this field is exploited only in the downlink frame (RFU in uplink), where the Network Server wants to inform the end-device to send as soon as possible an uplink frame (even empty) in order to have the possibility to send downlinks in the following receive windows.

– **FOptsLen** (4 bits): It specifies the length (0 to 15) of the FOpts field in the frame header of the MAC Payload. If 0 it means that no FOpts is present.

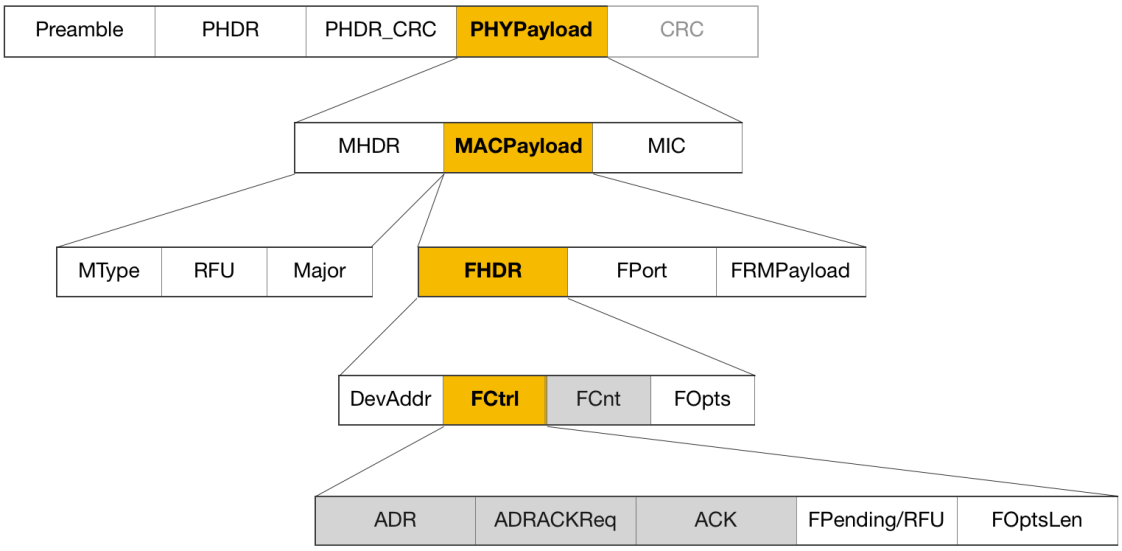


Figure 2.14: MAC message format structure. In gray the fields of interest.

2.3.4 MAC Commands

LoRaWAN offers several MAC commands to the Network Server and the end-devices, in order to adapt various physical layer parameters and to assess the status of devices or network links.

MAC commands, as shown in Figure 2.16, are sent with the aim of:

1. Adapting different physical parameters.

   - **LinkADRReq/Ans** for data rate and transmission power adaptation. With LinkADRReq, the network server requests the end-device to perform a rate adaptation. The payload has:
     – DataRate-TXPower (1 byte): 4 bit for DataRate, 4 bit for TX-Power.
     – ChMask (2 bytes): Channel mask. If one or more of the 16 bits is set to 1, it means that the corresponding channel can be used for uplink transmissions (if the data rate of the channel meets the current data rate of the end device).

– NbTrans (4 bits): Number of repetitions. It defined the number of times that an unconfirmed uplink has to be transmitted. This is done in order to guarantee a certain Quality of Service for a certain end-device uplink, without asking for an acknowledge. The default value is 1 and the maximum value is 15.
– ChMaskCntl (3 bits): Channel Mask control, it controls the interpretation of preceding ChMask. It is different from 0 if more than 16 channels are used in the network. Indeed, in certain regions, devices may have to store more than 16 channel definition (while the EU863-870 LoRaWAN only supports a maximum of 16 channels). When ChMaskCntl field is 0 the ChMask field individually enables/disables each of the 16 channels, thus, for EU863-870 it must be zero or the end-device will reject the command and unset the *Channel mask ACK* bit in its response. The LinkADRAns (1 byte) indicates the new status of the device (1 bit for Channel mask ACK, Data rate ACK, Power ACK each). A value of "1" in these fields states the success of the requested setting. If any of the three is "0" then the command did not succeed.

- **RXParamSetupReq/Ans** to change the default parameters of the end-device receive windows. RXParamSetupReq is sent in order to:
  – Set the RX1DROffset that allows programming an offset between the uplink data rate and the corresponding RX1 downlink slot data rate, as shown in Table 2.7. RX1 frequency can not be changed, and it is always equal to the one of the corresponding uplink message.
  – Change frequency and data rate (RX2DataRate) of the RX2 following each uplink. The default parameters for RX2 are 869.5MHz/DR0 (SF12, 125 kHz).
  It is important to highlight that if the user chooses to use parameters that differ from the default ones, it must communicate the decision to the Network Server – that can accept or refuse them – using an out-of-band channel during the end-device activation.
  With RXParamSetupAns, the end-device informs the Network Server if the setup was successful or not (since it could be unfeasible). The

Uplink DR	**Downlink DR in RX1 slot**					
DR0	DR0	DR0	DR0	DR0	DR0	DR0
DR1	DR1	DR0	DR0	DR0	DR0	DR0
DR2	DR2	DR1	DR0	DR0	DR0	DR0
DR3	DR3	DR2	DR1	DR0	DR0	DR0
DR4	DR4	DR3	DR2	DR1	DR0	DR0
DR5	DR5	DR4	DR3	DR2	DR1	DR0
DR6	DR6	DR5	DR4	DR3	DR2	DR1
DR7	DR7	DR6	DR5	DR4	DR3	DR2
	0	**1**	**2**	**3**	**4**	**5**
	RX1DROffeset					

Table 2.7: Downlink DR variation based on RX1DROffset.

RXParamSetupAns has a 1-byte payload indicating the device status: Channel ACK, RX2 Data Rate ACK and RX1DROffset ACK (all 1-bit fields) have the value of "1" if the requested parameters were successfully set, "0" otherwise.

- **NewChannelReq/Ans** to modify an existing channel or to create one. NewChannelReq is composed by:
  – ChIndex (1 byte): this is the index of the channel that has to be modified or created. Each end-device has to support at least 16 different channel definitions, where a channel definition corresponds to a frequency and a set of data rates usable on this frequency. Between these 16 channels, there are N default ones that could not be modified by the NewChannelReq. Thus, ChIndex can go from N to 16, since the default channel index range is 0 to N-1. In certain regions device may have to store more than 16 channel definitions (and this is how the Channel Mask Interpretation in the redundancy bit of the LinkADRReq command is used). In Europe N=3, so there are three default channels that correspond to 868.1, 868.3, and 868.5 MHz with DR0 to DR5 and they must be implemented in every end-device. Those default channels cannot be modified through the NewChannelReq command and they guarantee a minimal common channel set between end-devices and network gateways.
  – Frequency (3 bytes): 24 bits unsigned integer. The actual channel

frequency in Hz is 100xFreq. A value of 0 disables the channel.
– DrRange (1 byte): allowed data-rate range. 4 bits for MaxDR, 4 bits for MinDR. The minimum data rate subfield designates the lowest data rate allowed on this channel using the encoding specified in Table 2.1. NewChannelAns contains 1-bit fields to indicate a successful (value of "1") creation/modification of the channel.

- **RXTimingSetupReq** allows the configuration `RECEIVE_DELAY1`: the delay of the first reception window opening after the end of the uplink transmission. The delay, that can go from 1 (encoded with value 0 or 1) to 15. RXTimingSetupAns has no payload and it is just used to acknowledge the request.

2. Knowing the status of the device and of the network.

   - **LinkCheckReq/Ans** to validate its connectivity with the network. LinkCheckReq is used by the end-device to validate its connectivity with the network. In this case, the message has no payload. The server must respond with a downlink (LinkCheckAns) containing two fields of 1 byte each:
     – Margin: it indicates the margin in dB of the last LinkCheckedReq command that has been received by the server. It is an unsigned integer in the range of 0..254, where "0" means that the frame was received at the demodulation floor, while a value of "10", for example, means that the frame reached the last gateway 10 dB above the modulation floor.
     – GwCnt: number of gateways that have correctly received the LinkCheckReq command. This counter indicates how robust is the uplink transmission since a frame that is received by more than one gateway has a higher probability to successfully reach the server.
   - **DutyCycleReq/Ans** to limit the transmit duty cycle over all sub-bands. DutyCycleReq is a command used by the network coordinator in order to limit the maximum aggregate transmit duty cycle. The 1-byte payload of the request command is a value with valid range [0:15] and it is used in order to calculate the maximum end-

device transmit duty cycle allowed:

$$aggregateDutyCycle = \frac{1}{2^{MaxDCycle}} \tag{2.9}$$

If the aggregateDutyCycle has a value of "0", it means that there is not additional duty cycle limitation except the one indicated by the regional regulation. On the other hand, a value of "25"5 means that the end-device shall become immediately silent: it is equivalent to remotely switching off the end-device. DutyCycleAns does not contain any payload and it is used just to acknowledge the reception of the request.

- **DevStatusReq/Ans** to request the status of end-device. The Network Server can request the status of end-device by sending a downlink. The node response (DevStatusAns) is composed of 1 byte to indicate the battery state (a value of "1" indicates minimum power, while "254" the maximum) and 1 byte to state the demodulation signal-to-noise ratio in dB of the last successfully received DevStatusReq command.

A single data frame can contain any sequence of MAC commands, either piggybacked in the frame option field (FOpts) or, when sent as a separate data frame, in the FRMPayload field with the FPort field being set to 0. They are answered/acknowledged by the receiving end in the same order than they are transmitted. As a consequence, each time a MAC command is sent (either by the end-device or the server), a downlink is involved.

2.3.5 End-Devices Activation

To participate in a LoRaWAN Network, each end-device has to be personalized and activated. After the activation, at least the following parameters are stored in the end-device:

1. **End-device address** (DevAddr): 32 bit identifier within the current network. The seven most significant bits are necessary to establish the Network Identifier. Indeed, different network operators can present the same set of addresses in the same territory and it is necessary to be able

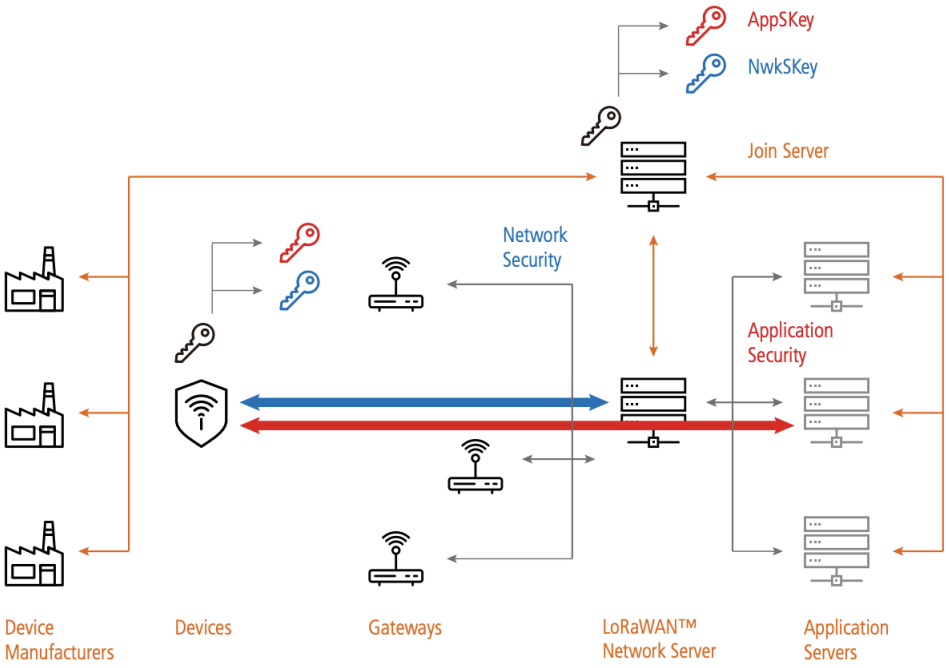


Figure 2.15: LoRaWAN security and keys, from [26].

to discern such overlapping addresses. The least significant 25 bits are used to identify the network address of the end-device, that is assigned by the network manager arbitrarily.

2. **Application identifier** (AppEUI): The AppEUI globally identifies the owner of the LoRa node. It is an ID in the IEEE EUI64 address space stored in the activation of the end-device before the activation procedure is executed.

3. **NwkSKey**: The NwkSKey is a network key that identifies a session and an end-device. The end-device and the Network Server use it with the aim of calculating and verifying the MIC (see Section 2.3.3) for all the received frames (that are encrypted and decrypted using such key) in order to guarantee data integrity, as shown in Figure 2.15.

4. **AppSKey**: The AppSKey is a key specific for an application and an end-device. The payload field of application-specific messages is encrypted and decrypted by the end-device and the Network Server using this key.

The activation can be performed by:

CID	COMMAND NAME	TRANSMITTED BY	STRUCTURE
0X02	LinkCheckReq	End-Device	No Payload
0x02	LinkCheckAns	Server	Margin (1) | GwCnt (1)
0x03	LinkADRReq	Server	DataRate_TXPower (1: DataRate 7:4, TxPower 3:0) | ChMask (2) | Redundancy (1: RFU 7, ChMaskCtrl 6:4, NbReq 3:0)
0x03	LinkADRRAns	End-Device	Status (1: RFU 7:3, PowerACK 2, DataRateACK 1, Channel Mask ACK 0)
0x04	DutyCycleReq	Server	MaxDCycle (1)
0X04	DutyCycleAns	End-Device	No Payload
0X05	RXParamSetup Req	Server	DL Setting (1: RFU 7, RX1DROffset 6:4, RX2DataRate 3:0) | Frequency (3)
0X05	RXParamSetup Ans	End-Device	Status (1: RFU 7:3, RX1DROffsetACK 2, RX2DataRateACK 0, Channel ACK 0)
0X06	DevStatusReq	Server	No Payload
0X06	DevStatusAns	End-Device	Battery (1) | Margin (1: RFU 7:6, Margin 5:0)
0X07	NewChannelReq	Server	ChIndex (1) | Freq (3) | DrRange (1: MaxDr 7:4, MinDr 3:0)
0X07	NewChannelAns	Server	Status (1: RFU 7:2, DataRangeOK 1, Frequency OK 0)
0X08	RXTimingSetup Req	Server	Setting (1: RFU 7:4, Delay 3:0)
0X08	RXTimingSetupAns	End-Device	No Payload

Figure 2.16: Most relevant LoRaWAN MAC Commands. In red MAC Commands sent through a downlink transmission.

- **Over-The-Air-Activation** (OTAA). This type of activation requires a join request and accept message. Before initiating the procedure, some information has to be given to the device:
  – The globally unique end-device identifier (DevEUI)
  – The application identifier (AppEUI)
  – An AES-128 application key (AppKey) assigned by the application provider (thus, exclusively known from the latter) and specific for the end-device. The AppKey is used to compute the session keys NwkSKey and AppSKey each time the end-device perform an OTAA.

Join Procedure

The join procedure is always initiated from the end-device by sending a *join-request uplink*. The network server will respond with a *join-accept downlink* if the end-device is permitted to join the network. The join-request message contains the AppEUI and DevEUI of the end-device followed by a random value (DevNonce). Each time an end-device tries to join the network, the server checks if the one of the DevNonce values stored in precedent sessions for the specif end-device is equal to the new one. The request is accepted only if the two values differ. Indeed, a reply attack can try to disconnect the current end-device by requesting multiple times to join the network. With the DevNonce it possible to prevent such a mechanism. It is important to highlight that the JoinRequest message transmit duty-cycle should never exceed 0.1%.

The network server will respond to the join-request message with a join-accept message if the end-device is permitted to join the network. The join-accept message is sent like a normal downlink but uses delays JOIN_ACCEPT_DELAY1 (default setting of 5s) or JOIN_ACCEPT_DELAY2 (6s) (instead of the usual RECEIVE_DELAY1 and RECEIVE_DELAY2, respectively). On the other hand, there is no change in terms of frequency or data rate with respect to the usual RX1 and RX2 receive windows.

The join-accept message contains:
– an application nonce (AppNonce). The AppNonce is a random value produced by the network server and utilized by the end-device to origi-

nate the two session keys NwkSKey and AppSKey, using its AppKey.
– a network identifier (NetID). The NetID is used to discern between different end-devices belonging to different networks. Also, it contains some variable useful information that the network operator can arbitrarily choose to incorporate (such as the DevAddr or the RxDelay, defined ad a delay between transmission and reception).

- **Activation By Personalization** (ABP). Under certain circumstances, end-devices can be activated by personalization. Activation by personalization directly ties an end-device to a specific network by-passing the join-request/join-accept procedure. In this case, The end-device is already equipped with the required information for participating in a specific LoRa network when started, by storing DevAddr and the two session keys NwkSKey and AppSKey directly into the end-device.

Chapter 3

Downlink Traffic Problematics

Based on the in LoRaWAN architecture previously described it is possible to identify two types of streams:

- **Uplink traffic** is the traffic that flows from the end-devices (ED) towards the Network Server (NS).
- **Downlink traffic** is the traffic that flows in the opposite direction.

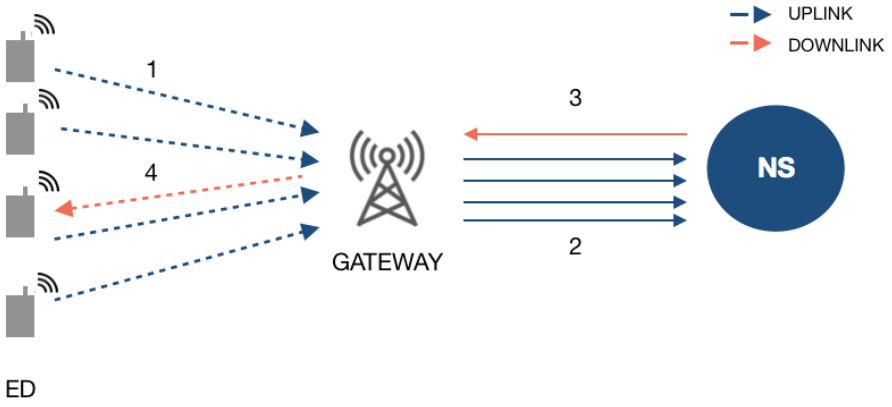


Figure 3.1: LoRaWAN streams.

In a nutshell, downlink frames arise for a variety of reasons:

(a) Reception of Confirmed Messages (CONF) and subsequent ACKs;

(b) Management functions required by LoRaWAN, e.g. the OTAA join procedure for devices;

(c) MAC commands for parameters adaptation and link connectivity testing, e.g. LinkADRReq, LinkCheckAns, etc.;

(d) In response to frames with ADRACKReq bit set;

(e) Events notification from IoT applications (e.g. software update, sampling interval change, ...).

In this kind of network, pure uplink traffic, from end-devices to gateways, is strongly favored. Typically, thousands of battery-powered devices (e.g. sensors) are sending small and sporadic messages to a server. For this reason, the existing literature focuses mostly on IoT best-effort applications, exhibiting scenarios that show only the presence of Unconfirmed Messages (UNC), without taking into account downlink traffic even when it would have been necessary (e.g [15] [30] [42]). This is usually done with the purpose of improving parameters such as the Packet Delivery Rate (PDR). In this context, the loss of some packets has marginal importance, since the main focus is just to collect the largest possible amount of data to enable the data analysis or prediction. However, the huge set of possible application domains makes this assumption just marginally true, since different applications require different QoS [11]. An IoT environmental data collection system (such as the one studied in [36] for smart agriculture), require a minimal QoS. The sensor main task is to send real-time local environmental data (e.g. air temperature, humidity, wind speed, etc..) to cloud storage with the aim of enabling the data analysis by a remote computer and the deployment of a machine-learning algorithm to perform some environmental predictions. Here, the loss of some packets is negligible. On the other hand, for a forest fire prevention application [17], the loss of a packet signaling smoke or fire is of critical importance and its reception necessarily needs to be acknowledged.
The requirement of downlinks to assist a certain QoS is also clear in smart city applications engineered to enable smart lighting and smart metering, requiring the confirmation of commands sent to the nodes. In this context, LoRaWAN seems to be an appealing choice thanks to its low power consumption characteristic: changing battery could be a problematic task to perform in several applications and decreasing the need for such operation is an important requirement. Moreover, some studies have proven the possibility to combine the

traffic monitoring with the vibrations generated by vehicles to support the energy consumption of LoRa end-devices [47].

Wireless Sensor Networks create the possibility of exploiting the IoT for mission-critical applications, like health care solutions or natural disaster prevention. Sensors, enabling instant reaction to any alarm, can provide adequate situational awareness and decision-making support to manage crisis situations [58]. LoRaWAN, enabling low-power consumption and long-range communication, could be considered as an ideal candidate to support the requirement of IoT critical applications. For this reason, recently the focus of researchers studying critical application deployment is shifting towards this technology [3]. These applications result to be particularly sensitive to packet loss when sending packets of critical importance [45] (e.g. application monitoring fall detection [5]), whereas can show a certain fault tolerance when sending updates (e.g. when monitoring glucose levels or heart rhythms). The detection of elderly people falls is performed thanks to the usage of an accelerometer sensor attached to the patient's thigh. The sensor detects any change in the rotation and can predict a fall (in this case the data is locally elaborated) or detect one. In this case, the signal must be transferred to the health care providers and the application shows a strict QoS requirement.
Catherwood et al. in [8], examine the use of LoRa for Urinary Tract Infection (UTI) monitoring and diagnosis. The main attractiveness of the technology is identified in the absence of Wi-Fi, subscription fees or SIM cards or access to the 3G/4G mobile network and the patient's test results are acknowledged to ensure the capture.
The deployment of LoRaWAN for health care application has been evaluated by Baker et al. in [22]. For the most part, LoRaWAN is reasonably well-suited thanks to its range, latency, and network capacity taking into account that only critical data (such as a heart attack) will require the maximum reliability.

Even if multiple QoS service metrics must be met when deploying an IoT application, this thesis focuses mostly on just one QoS criteria: reliability. Most of these kinds of applications, based on the need to immediate response, require the sending of high-priority traffic on a reliable data transport system

[55]. LoRaWAN implements a data retransmission scheme in order to guarantee transmission reliability, and the usage of CONFs combined with ACKs is the method deployed to assess the necessity of retransmission in case of data loss. Nevertheless, the use of CONFs is currently not recommended due to its negative impact on the network. But even in the absence of CONFs, the presence of occasional downlink traffic is guaranteed by for reasons (b), (c), (d), and possibly (e). Hence, it is important to understand which are the problems related to downlink frames along with the normal uplink traffic and how this affects network capacity. Moreover, improving downlink scalability can enable the engineering of IoT LoRaWAN applications that require such reliable data communication [1].

3.1 Problem Inspection

During the downlink problematics analysis, I established three main causes related to the negative impact of downlinks in the network. The following sections aim at offering a clear explanation for the above-mentioned issues.

3.1.1 Duty-cycle Saturation

As shown in Chapter 2, LoRaWAN is a MAC protocol implemented on the LoRa modulation in the ISM band. The use of the spectral radio frequencies in the ISM band is regulated by the ETSI standard [54]. In order to be compliant with the standard, the current LoRaWAN specification exclusively uses duty-cycled limited transmissions with an ALOHA-type access method, applying a sub-band limitation. In Europe, LoRaWAN operates in the 863-870MHz band, for which the standard defines the following sub-bands and relative duty cycle:

- g (863.0 - 868.0 MHz): 1%
- g1 (868.0 - 868.6 MHz): 1%
- g2 (868.7 - 869.2 MHz): 0.1%
- g3 (869.4 - 869.65 MHz): 10%
- g4 (869.7 - 870.0 MHz): 1%

In those bands, the duty cycle ratios (DutyCyclesubband) range from 0.1% to 10%. Each time a frame is transmitted, its emission timestamp and its Time On Air (ToA) are recorded by the emitter for each transmission sub-band. The ToA defines the transmission time of a frame and it depends on the frame size, SF, BW, and CR (as shown in the subsection 2.2.1).
Each time a frame is transmitted, the emission timestamp and the Time On Air (ToA) are registered for the sub-band used for the transmission. The same sub-band can not be used again for the subsequent time off (Toff) seconds, where

$$TOff_{subband} = \frac{TimeOnAir}{DutyCycle_{subband}} - TimeOnAir \tag{3.1}$$

Example. A device transmits a 0.5 s long frame on a channel belonging to sub-band g1 characterized by 1% of duty-cycle. As a consequence, the whole sub-band (868 - 868.6) is saturated, since it is not available for the succeeding 49.5 s.

This limitation must be respected by both the end-devices and the gateways. In this context, it is evident that, in scenarios with downlink traffic, the gateways can potentially become the bottlenecks of the network, by saturating the duty-cycle and therefore by exhausting their ability to forward downlinks to the end-devices.

Taking into consideration class A end-devices (the most used), the gateway must send the downlink exactly at the begging of one of the two receive windows in order to allow the end-device' s radio receiver to detect the downlink's preamble. If the duty-cycle is saturated for both the sub-bands corresponding to the transmission channels of RX1 and RX2, the gateway will not be able to forward the downlink in neither of the two and the end-device will perform the retransmission of the uplink for `NbTrans` additional times, without receiving an ACK (the NS will never send an acknowledge twice). It is worth highlighting that a confirmed message is considered lost if it does not receive the ACK, even if the uplink has been correctly received by the Network Server. In this case, the end-device will consider the missed ACK as a clue of bad link connectivity and will eventually try to switch to a lower DR, even if it would have not been necessary.

It is also important to highlight that switching to higher spreading factors

produces, at every single increment, the duplication of the Time On Air. This is particularly meaningful for the downlink transmission performed in RX2, which is characterized by a default spreading factor of 12 and, as a consequence, a longer Time off for the sub-band.

3.1.2 Gateway's Half-duplex Nature

Due to the lack of frequency band separation between uplink and downlink communication, LoRa gateways operate in half-duplex mode, i.e. they cannot receive and transmit at the same time and the communication is one direction at a time. When a packet is sent, all the ongoing receptions are aborted and no uplink can be received for all the duration of the downlink Time On Air.

The presence of downstream traffic in large networks will consequently bring to an overall decrease in the end-devices PDR.

Finally, it is possible to identify two different modes related to the gateway ability to receive packets:

- the gateway is "*free*" when the simplex channel is used for receiving uplinks. In this period the gateway is receptive and all the uplinks received are forwarded to the Network Server.

- the gateway is "*busy*" when the transmission direction is switched and the gateway starts sending a downlink. When in this mode, the gateway is unable to receive any uplinks.

3.1.3 Downlink Frames Sequential Sending

The LoRa concentrator can have only one TX packet programmed for departure at a time [34]. The actual departure of a downlink packet is performed based on its timestamp when the concentrator internal counter reaches the timestamp's value. The modulation profile (SF, CR, BW, ..) and the emission timestamp of a downlink frame are computed by the Network Server at the downlink frame scheduling phase.

Thus, downlink frames are buffered in a Just In Time queue and sent exclusively sequentially (while the simultaneous reception of several uplinks at the same time is guaranteed by the presence of two transceivers). This hardware

constraint exacerbates the problem caused by the gateway half-duplex mode and speeds up the duty-cycle saturation process for the downstream traffic.

3.2 Example Scenario of Downlink Problems

In order to have a comprehensive vision of the problems previously exposed, this example shows a basic scenario where it is possible to observe all of them together.

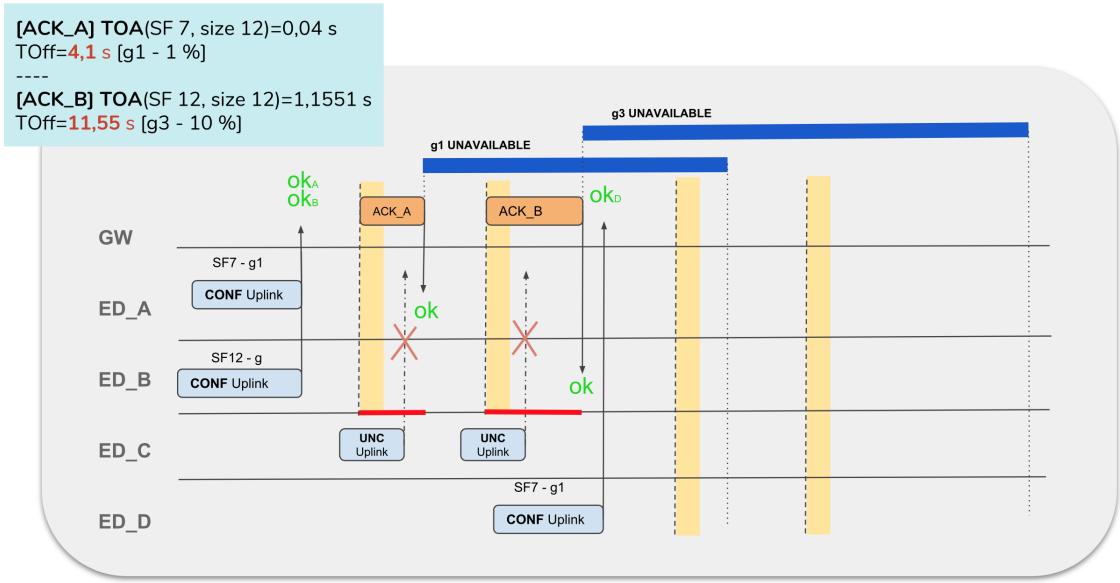


Figure 3.2: Downlink traffic problems.

We consider a network with one server (not shown for the sake of simplicity), one gateway (GW) and four end-devices (ED). The following events follow each other in time:

1. ED_A sends a CONF on a channel belonging to sub-band g1 with spreading factor 7. ED_B sends a CONF on g with SF12. ED_A and ED_B will open RX1 at the same time.

2. The two uplinks are received by the NS. It is in charge of the scheduling of the ACK (generally 12 bytes long) for both of them. Since it is possible to transmit just one downlink at the same time, the NS decides to send first ACK_A in RX1 (sub-band g1, SF7) and then ACK_B in RX2 (sub-band g3, SF12).

3. During the sending of the two downlinks, ED_C sends two packets. Both of them are not received by the NS due to the half-duplex mode of the gateway. The *no-reception window* of the gateway is highlighted in red. All the packets supposed to be received in this time window are lost.

4. After the sending of ACK_B, ED_D sends a CONF on a channel of the sub-band g1. The server receives the uplink, but it can not schedule an ACK since the gateway is not able to forward it neither in RX1 (sub-band g1 saturated) nor in RX2 (sub-band g3 saturated).

It is worth highlighting that the two RX1s in the first step do not necessarily have to fall exactly at the same instant: the problem would show up even if the two downlink transmissions slightly overlap.

Chapter 4

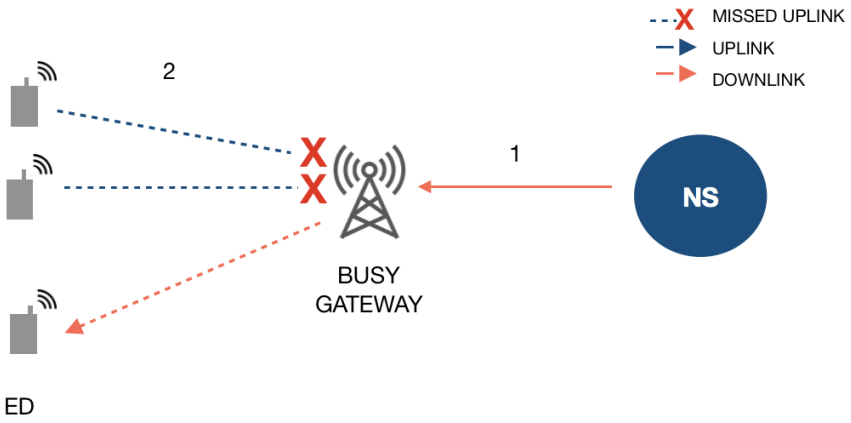
Proposed Solutions

Different strategies can be explored in order to face the problems observed in the network when downlink traffic is introduced. Three different solutions will be detailed in the next sections. Each of them is meant to handle one of the problems previously exposed.

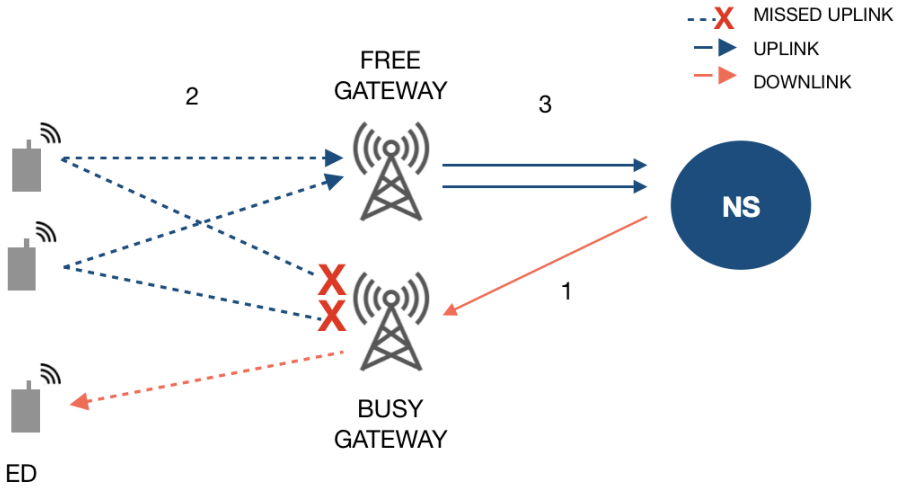
4.1 Multi-gateway

A possible solution, able to smooth the problems caused by the gateways half-duplex mode, consists of a different network architecture implementation. Indeed, by deploying additional gateways, the uplinks that are supposed to be received during the transmission of a downlink, could still be received by all the other gateways that are not transmitting.

In this scenario, the *no-reception window* of a gateway is covered by all the other "free" gateways, which are in charge of receiving the uplinks missed by the "busy" gateway and of forwarding them to the Network Server, as shown in Figure 4.1b. The latter will then handle the de-duplication process.



(a) Single-gateway configuration. When the gateway is busy sending a downlink, all the oncoming uplinks are missed.



(b) Two-gateways configuration. The uplinks are received and forwarded to the NS by the free gateway.

Figure 4.1: Uplink reception in a single (a) and multi-gateway (b) network architecture configurations.

4.2 Balanced Gateway Selection Algorithm

When the Network Server needs to send downlink traffic to an end-device, it has to select the gateway that will carry on the downlink transmission. Indeed, in a multi-gateways scenario, the Network Server could possibly receive the same uplink more than once, if it is received and forwarded by different gateways.

Currently, the LoRaWAN specifications do not provide any pieces of information or recommendation on how the Network Server should select the gateway. One possible selection algorithm, adopted by an open-source LoRaWAN network-server (the LoRaServer project ™[1]), is based on the Signal to Noise Ratio (SNR), a measure that compares the level of a signal to the level of background noise. For this reason, it is considered a useful parameter to estimate the wireless link quality [60].

In the SNR-based algorithm, the selection just depends on the information carried by the last packets received from the targeted end-device: in the de-duplication phase the server stores the uplink received with the best SNR - and after a certain threshold, the best Received Signal Strength Indication -, notes the gateway who forwarded it, and discards the other duplicates.

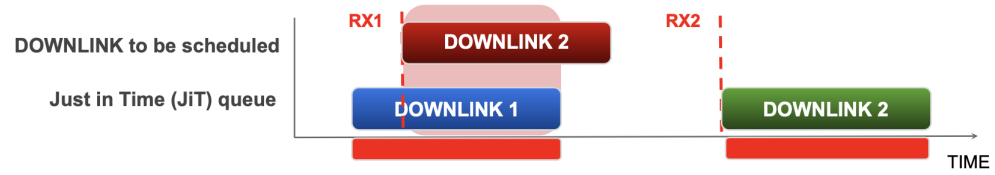
It worth noting that a downlink sent to a gateway with the best SNR but with the requested sub-band saturated, surely will not be forwarded to the end-device. As highlighted in subsection , the duty-cycle of a gateway that serves a large number of end-devices is quickly saturated and the gateway will often miss the receive windows opened by the motes. On the other hand, if the selected gateway is consistently receiving a big amount of upstream traffic, the number of missed uplinks will be relevant during the transmission of the downlink. Thus, with the current selection algorithm, the load of the gateway is not taken into consideration and the packets' loss will surely be consistent.

The downstream traffic could be better balanced and spread through the network if the gateway selection algorithm would also include the gateway's duty-cycle saturation of the requested downlink channel. In this context, the selected gateway could not be the best in terms of SNR, but the downlink will certainly have a higher probability to be received by end-device compared with the previous solution, where it would have not even been sent.

4.3 Parallel Sending and Downlink Combination

The current hardware design of LoRa gateways allows the transmission of a single frame at a time. Nevertheless, it is possible to imagine a hardware de-

[1] `https://www.loraserver.io`



(a) Sequential sending configuration

(b) Parallel sending configuration

Figure 4.2: (a) Possible configuration for a new downlink (2) to be scheduled in a gateway with sequential sending. It collides an already scheduled downlink, and it is scheduled in its RX2. (b) Enabling the parallel sending, the downlink is scheduled in its RX1 and the *no-reception window* (in red) is sensitively shorter.

sign able to overcome this restraint, since it is not inherent the technology. The idea here is to not only allow frames emissions in parallel on different channels but also simultaneously on the same channel with different orthogonal spreading factors, similar to what is done in UMTS cellular networks. Indeed, if LoRaWAN gateways had the ability to send more downlink frames simultaneously, the *no-reception window* and the TOffsubband would improve considerably.

Two factors have to be taken into account in the implementation of the downlinks parallel sending:

- Downlink frames combination requires that the receive windows of both candidate receivers overlap. This can be made more likely by lengthening the RX duration, as explored by Centerano and Vangelista [9]

- The gateway must still respect the duty-cycle limitation. It means that downlink can be sent in parallel just if in different sub-bands. Nevertheless, it is possible to imagine a downlinks combination where more downlinks on the same channel with different spreading factors are combined together, and then correctly demodulated by the relative end-devices.

- The gateway must consider also the restriction on the maximum Effective Radiated Power (ERP) imposed by the ETSI regulation. All the sub-bands used by LoRa are limited to a maximum radiated power of 14 dBm, except for the default sub-band of RX2, characterized by an ERP limitation of 27 dBm. As a consequence, to stay within the ERP limitation, it is necessary to compute a correct transmission power profile when combining several downlink frames on the same sub-band.

4.4 Example Scenario of Proposed Solutions

In order to perceive the impact of the proposed solutions on the network, the same scenario shown in subsection 3.1.4 is presented.

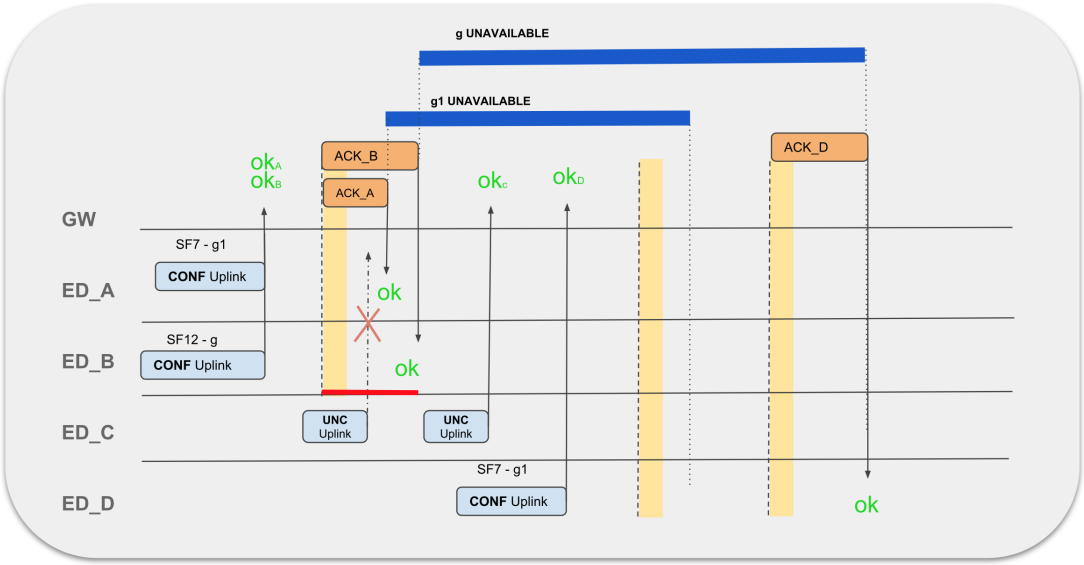


Figure 4.3: Downlink traffic solutions implementation.

1. ED_A and ED_B perform the same operations described in the previous example.

2. This time, thanks to the parallel sending feature, the Network Server is able to send both ACKs for ED_A and ED_B in RX1, using the sub-bands of the respective uplinks (g1 for ACK_A, g for ACK_B).

3. ED_C sends two packets. The first one is still not received by the gateway since it is transmitted while the gateway is busy sending the downlinks.

In a multi-gateways scenario, the same packet can still reach other "free" gateways and hence the Network Server will receive it.

In this case, unlike the previous one, the second uplink sent by ED_C is correctly received by the gateway since it is not busy anymore. Thanks to the parallel sending, the *no-reception window* is considerably shorter: if with the sequential sending it consisted of the sum the two ToA (ToA(ACK_A)+ToA(ACK_B)), now it is just the longest ToA (in this example the one of ACK_B).

4. ED_D sends a CONF on a channel belonging to the sub-band g1. The Network Server is still not able to send ACK_D in RX1 (g1 saturated), but, differently from the previous case, g3 is not saturated and the ACK can be sent in RX2.

   Thanks to the parallel sending, the duty-cycle saturation process arises for much higher loads.

4.5 Related Work

Downlink traffic is arousing interest only recently. Most of the studies on the topic principally focus on showing its negative impact on the network, usually identifying the duty-cycle as the main or single problem.

Adelantado et al. [21] conduct an analysis of LoRaWAN in order to identify the main problems of the technology. Among other issues, the duty-cycle saturation is recognized as one of the limits of LoRaWAN, especially for the development of ultra-reliable applications which, as such, require acknowledgments.

Pop et al. [32] address the bidirectional traffic problem by introducing LoRaWANSim simulator, an extension of LoRaSim that includes downlink frames and retransmissions. Some of the findings are relative to the aggressive data-rate back-off approach during retransmission recommended in the old version of LoRaWAN specification, and hence they are not relevant anymore. On the other hand, they correctly identify the scalability problems that arise when a large number of nodes request ACKs and the duty-cycle limitation issue.

An empirical evaluation of the downlink traffic impact is conducted by Mikhaylov et al. [16]. The results show the strong negative effect on the performance of uplink frames, consistent with the previous studies. Nevertheless, the explanation behind this phenomenon does not take into account the half-duplex nature of the gateway and the I/Q inversion performed by gateways when transmitting uplink and downlink frames, in order to avoid their collision.

The aim of Centenaro et al. [25] is to investigate the impact of downlink frame feedback in a multi-gateway architecture using an event-driven simulator. In order to have some scheduling flexibility on the network side, one of the modifications of the model is to have received windows one second longer than the actual ones. This significant change imposes that each RX window stays open for an unreasonable amount of time and the resulting downlink frame scheduling process is completely different from the real one. However, the obtained results show again the negative impact of downlink frames without providing additional reasons besides the duty-cycle issue.

One of the main contributions to the downlink frame issue is given by Van et al. [20] by performing a downlink traffic analysis using the ns-3 network simulator. The authors clearly explain the reasons for gateways congestion by highlighting the duty-cycle limitation, whereas, the half-duplex problem is merely mentioned. A multi-gateway architecture is proposed and improvements are evaluated in terms of lower duty-cycle saturation and better distribution of the workload through the gateways. Yet, neither the explanation nor the proposed solution takes into consideration the gateway selection algorithm. In their implementation, the server simply tries to schedule an ACK in the first available gateway among the pertinent gateway(s).

Most part of the analyzed papers miss a clear and well-detailed explanation of the reasons behind the downstream issues and show a scarcity of significant and specific solutions to solve them.

This work aims at providing these two lacking components.

Chapter 5

Simulation Data and Evaluation Approach

In the first experimental phase, the LoRaServer Project has been exploited combined with the deployment of several LoRa®PicoCell Gateways [64] and four Adeunis LoRaWAN Field Test Device [38]. The LoRaServer Project network architecture is composed by LoRa gateways that forward the received data to the LoRaWAN network-server publishing it with MQTT via the ChirpStack Gateway Bridge, as shown in Figure 5.1). This server is able to perform the OTAA activation procedure and sends downlinks when necessary.

The main objective of these experiments was to observe how real end-devices and server would behave during the join procedure and how they would respond when changing the several LoRa parameters through MAC commands. The devices were registered to the LoRaServer and the payload length, spreading factor, and transmission power is changed and evaluated with dynamically varying radio conditions. The most valuable information gained through these experiments has been about the physical parameters set for the downlink generation (such as the payload length of a MAC command response). The server does not check the duty-cycle limitation and schedule the downlink in the gateway that forwarded the packet with the best SNR value.

The PicoCell Gateway is a multi-channel transceiver designed to simultaneously receive several LoRa packets using different spreading factors. The RF front-end architecture of the PicoCell GW is characterized by a half-duplex

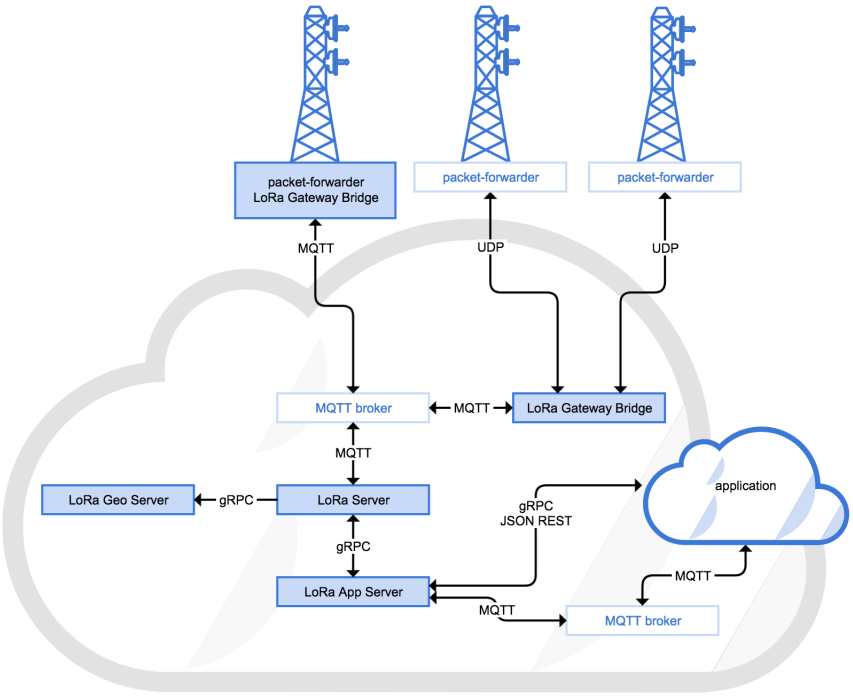


Figure 5.1: LoRaServer Project network architecture.

mode and a maximum transmits output power of +20dBm. The two SX1257 transceivers [18] shown in Figure 5.2 are highly integrated RF front-end to digital I and Q modulator and demodulator. Two transceivers are used instead of one to be able to simultaneously receive 8 LoRa channels multi-data rates.

As highlighted before, the limits of this technology become significant in large-scale networks. Nevertheless, in order to produce experimental results able to validate the proposed solutions, it would have been necessary to deploy not just some gateways and a Network Server, but also a considerable number of end-devices. Indeed, the Network Server is able to interact (e.g. sending downlinks) just with registered end-devices, while all the traffic received by non-registered devices is systematically ignored. Despite the availability of the LoRaServer and of several PicoCell Gateways, it was not possible to provide a number of end-devices significant enough to create a real network deployment able to show the problems that arise when downlinks are required. For this reason, in order to evaluate and quantify the improvements produced by the proposed solutions, it was chosen the pragmatic approach to implement a simulator that exploits real traffic traces to observe the behavior of the net-

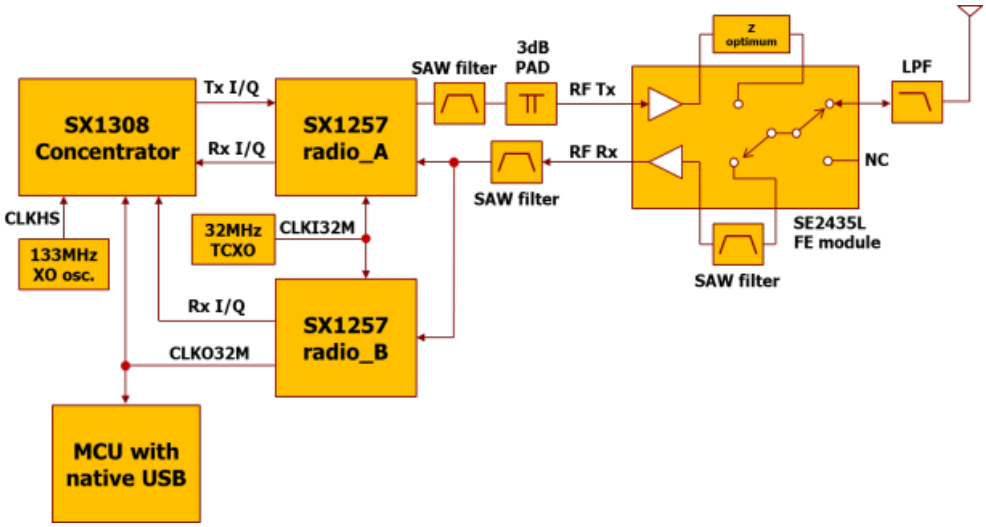


Figure 5.2: PicoCell Gateway RF Block Diagram.

work when it is virtually exposed to various percentages of downlink frames, relatively to the uplink traffic.

5.1 Real Traffic Traces Collection

Instead of having a traffic generation model that follows a Poisson distribution ([23] [24] [27]), the idea of exploiting real traffic was more compelling and valuable. To obtain such collection, four Semtech LoRa PicoCell Gateways were set up and attached to Raspberry Pi boards. The WalT platform greatly facilitated the deployment.

5.1.1 Experiment Set up

In order to collect real traffic traces, four gateways have been deployed. The position of the gateways is meant to produce a configuration where a gateway serves a large number of nodes and at least part of the traffic received by it is shared with the other gateways (in order to have some duplicates), as shown in Table 5.1.

All the gateways have been positioned in an urban scenario at the fourth floor of the IMAG building at the *Université Grenoble Alpes*, but in different conditions (as shown in Figure 5.3) to differentiate the traffic load:

- **GW1**. The main gateway has been positioned on a balcony, in order to provide the maximum collection of traffic.
- **GW2**. The second gateway is positioned outside of a window near the balcony. This gateway will receive almost the same traffic as GW1.
- **GW3**. The third gateway has an indoor displacement, in a room on the same building side of GW1 and GW2.
- **GW4**. The fourth gateway is located in a room on the other side of the building.

Table 5.1: Gateways set up for real traces collection.

GW	Displacement	Received traffic
1	Outdoor	100%
2	Outdoor	52%
3	Indoor	6,5%
4	Indoor	4,7%

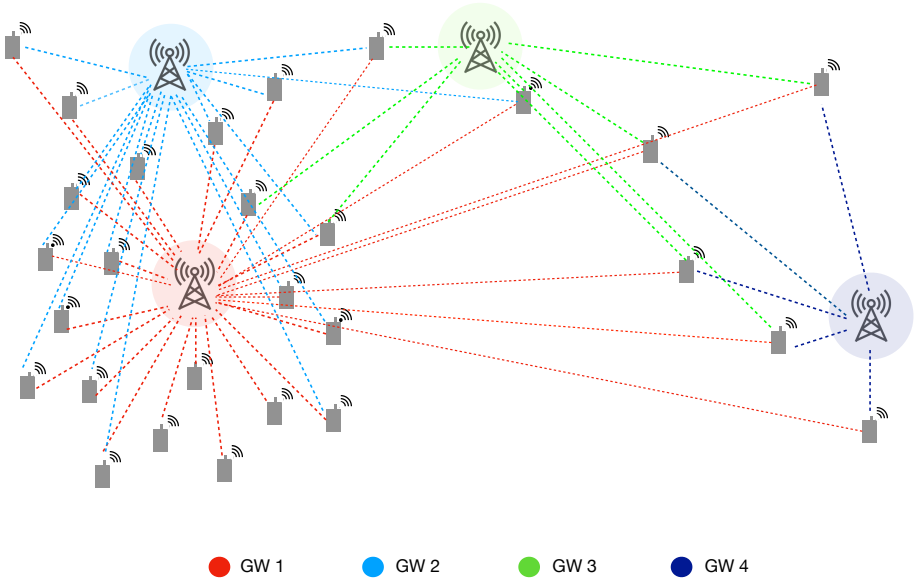


Figure 5.3: Hypothetical configuration of the experiment set up. GW1 is receiving packets from all the registered node, GW2 is receiving most of them, GW3 and GW4 are receiving a minimal part of the traffic.

The traffic collection has been performed several times, on different days, with different duration windows and weather conditions. In order to choose the best trace, the main parameter has been the number of packets received in the given time by all the gateways. Indeed, more packets allow the recreation of a more realistic scenario, where thousands of end-devices are communicating with the gateways in their radio range. In terms of duration, it has been observed how one hour of collected data result to be the ideal threshold time to incorporate a good representation of a real network deployment: with smaller durations, the possibility to observe the negative effect of downlink (for example when changing data rate) was lower, while longer duration reproduced very similar results.

In the selected trace, the main gateway received 5368 packets from 34 motes. Even if the number of end-devices is not into the hundreds, the captured traffic is intense enough to saturate the downlink channels, as long as enough frames are virtually changed to confirmed. All the 34 end-devices are considered registered with the server, that will treat their packets as if they were addressed to its application. The end-devices are considered already registered since it was not possible to recreate the join procedure characteristic of the OTAA. Indeed, we do not have any control over these end-devices, that can send packets of different sizes, DR and SF. Their behavior can just be changed afterward the capture, by manipulating the data received and replacing the "ACK" bit to 1, in order to force a downlink from the server.

To feed the simulator with the collected traffic traces, the latter is stored in a comma-separated-value file where each line represents a packet received by a gateway. To perform such operation, it has been necessary to modify the software running on the gateway (see subsection 5.1.2). The file, sorted by packets arrival time, represents the traffic that a server would receive in a given period of time.

5.1.2 Packet Forwarder Software

The packet forwarder is a program running on the host of a LoRa gateway that forwards RF packets received by the concentrator to a server through an IP/UDP link and passes to the PicoCell the RF packets that are sent by the server for the end-devices. Each task is performed by a different thread:

Listing 5.1: Structure containing the metadata of a packet received by the gateway's concentrator.

```
struct lgw_pkt_rx_s {
    uint32_t    freq_hz;
    /*!> central frequency of the IF chain */
    uint8_t     if_chain;
    /*!> by which IF chain was packet received */
    uint8_t     status;
    /*!> status of the received packet */
    uint32_t    count_us;
    /*!> internal concentrator counter for timestamping, 1 microsecond
     resolution */
    uint8_t     rf_chain;
    /*!> through which RF chain the packet was received */
    uint8_t     modulation;
    /*!> modulation used by the packet */
    uint8_t     bandwidth;
    /*!> modulation bandwidth (LoRa only) */
    uint32_t    datarate;
    /*!> RX datarate of the packet (SF for LoRa) */
    uint8_t     coderate;
    /*!> error-correcting code of the packet (LoRa only) */
    float       rssi;
    /*!> average packet RSSI in dB */
    float       snr;
    /*!> average packet SNR, in dB (LoRa only) */
    float       snr_min;
    /*!> minimum packet SNR, in dB (LoRa only) */
    float       snr_max;
    /*!> maximum packet SNR, in dB (LoRa only) */
    uint16_t    crc;
    /*!> CRC that was received in the payload */
    uint16_t    size;
    /*!> payload size in bytes */
    uint8_t     payload[256];
    /*!> buffer containing the payload */
};
```

- **tread_up** receives packets and forwards them to the server. It cyclically calls a non-blocking function (defined in the LoRa concentrator Hardware Abstraction Layer) that will fetch up to 'max_pkt' packets from the LoRa concentrator FIFO and data buffer and it processes the inbound packets and metadata stored in an array of fixed dimension (NB_PKT_MAX = 15) of RX packets. The struct of an incoming packet is shown in Listing 5.1, and it is used to serialize Lora packets metadata and payload.

- **thread_down** polls the server and enqueues packets in a Just In Time (JIT) queue. It prepares the downlink struct before checking its validity in terms of unsupported frequency or transmission power (also, if it is

an ACK, it cannot be a duplicate or out-of-sync).

- **thread_jit** checks the packets to be sent from the JIT queue and sends them by transferring data and metadata to the concentrator, and schedule transmission. It shares with *thread_down* the *jit_queue*, where all the enqueued TX packet struct (similar to Listing 5.1) are stored.

The program has been modified in order to accept as input a unique ID number to identify a gateway and to write in a CSV file each time a packet is received. This is done by making *thread_up* writing a file using the metadata about the current packet already stored in the RX packet struct used for serialization. The CSV file header is made of 16 fields:

- **GW_ID**. A unique number that identifies the gateway. This is assigned arbitrarily each time the packet forwarder software is launched.
- **PKT_ID**. A sequential number that identifies each packet.
- **SEC, MICROS**. Seconds and microseconds in UNIX Epoch time of the machine at the arrival of the packet. This information is used in the sorting phase as the common time reference between different gateways.
- **TMSTMP**. An internal timestamp of the gateway concentrator.
- **MODE**. Char used to identify the packet mode. 'U' stands for Unconfirmed, 'C' for confirmed, 'D' for duplicate. In this phase, all the packets are registered as unconfirmed.
- **MOTE**. A unique end-device identifier for the current session (it changes each time the device is turned off).
- **Packet Radio Informations**:
  - FCNT, number of uplinks sent without having received a downlink
  - SIZE, in bytes
  - SF, SNR, RSSI
  - CH, number in the range 0 to 8 that identifies a channel. The codification of it is defined in the gateway's config file.

- BW, 125 or 250
- FREQ, the frequency used by the packet
- CR, the code rate expressed as an integer

At the end of the experiment, each gateway has produced the CSV file containing the metadata of all the received packets. To simulate the real traffic that a server would receive from the different gateways, it is necessary to merge all the CSV files in one, where the packets are ordered on arrival time. Figure 5.4 shows how two packets (sent from mote 0000710B and 000157C3) are differently received from the four gateways, and how multiple receptions produces duplicates. The first packet is just received by gateway 1 and gateway 2.

GW	ID	SEC	MICROS	TMSTMP	MODE	MOTE	FCNT	SIZE	SF	BW	SNR	RSSI	CH	FREQ	CR
2	138	1531218473	291	103311060	U	0000710B	52039	29	7	125	1.8	-104	3	867.1	1
1	172	1531218473	312	136424492	U	0000710B	52039	29	7	125	4	-94	3	867.1	1
4	6	1531218473	802	59937756	U	000157C3	1901	17	12	125	-18.2	-100	0	868.1	1
3	8	1531218473	803	78774172	U	000157C3	1901	17	12	125	-6	-97	0	868.1	1
2	139	1531218473	813	103777108	U	000157C3	1901	17	12	125	7.5	-111	0	868.1	1
1	173	1531218473	829	136890540	U	000157C3	1901	17	12	125	4.5	-105	0	868.1	1

Figure 5.4: Some lines of a file trace produced by merging the ones of four gateways.

5.1.3 WalT Platform

The deployment of a specialized platform (like IoT-LAB or ORBIT) to perform the experimental phase has been initially evaluated. The principal advantage offered by this kind of platform is the assistance for experiments repeatability and the serialization of the experimental task. One of the main requirements for our experiments, in order to produce meaningful results, was to reproduce in the most accurate way possible a real LoRaWAN architecture with realistic radio parameters. With these intentions, a specialized platform fails to fulfill its purpose since nodes position differs from a real-world deployment (they are usually inside of a large room and displaced in a uniform grid) and the environmental conditions are surely static, being completely artificial.

Another option that has been evaluated is the WalT platform. WalT is a cheap, reproducible and highly configurable platform for network experiments

[43]. It provides full remote control over nodes (e.g. rebooting, remote shell sessions, etc.). WalT nodes are single-board computers running the OS (filesystem, kernel) wrapped in a docker image. The OS image can be cloned from the docker hub, easily customized and published to facilitate the experiments sharing and reproducibility, as shown in Figure 5.5. For these reasons, WalT is resulted to be the most appropriate platform to conduct the experimental phase.

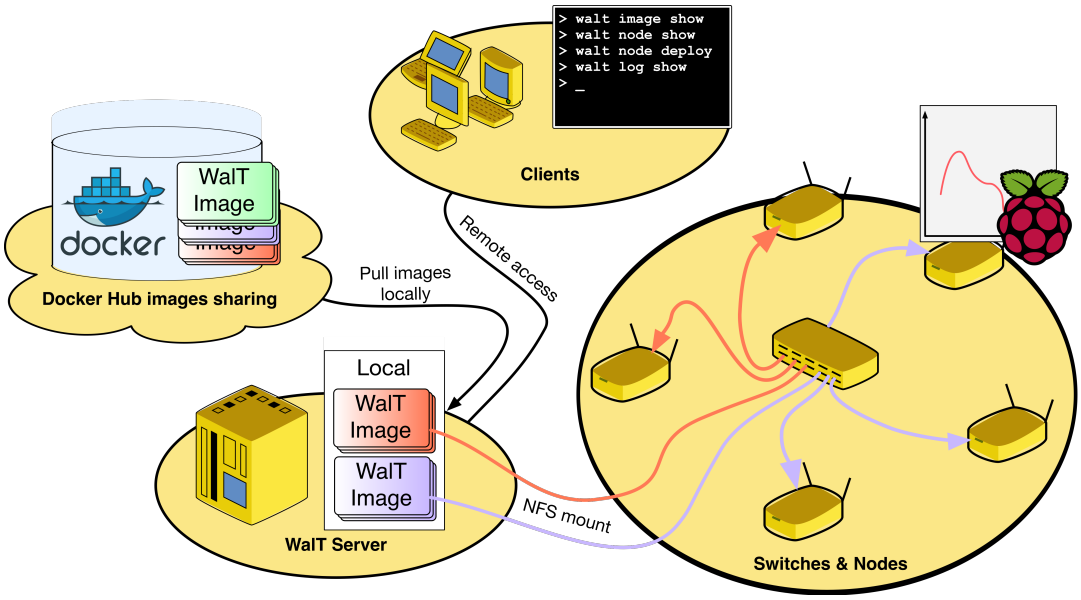


Figure 5.5: WalT architecture.

In the experiment setup, each Raspberry Pi used to implement a LoRa PicoCell Gateway is a WalT node. In order to accommodate the need of hosting the modified Packet Forwarder Software that runs on each node, the Raspberry Pi's original operating system image is edited and permanently stored in the Docker Hub. Each node is booted with the new image.

Such modification in the software enables the production, by each gateway, of a traffic trace that can be combined with those of other gateways, by sorting the packets by UNIX Epoch time. Obviously, all the traces need to be captured in the same time window. In this way, it is possible to create different traces combinations based on the number of gateways we want to deploy in the simulation.

One of the main problems encountered during the execution of the experiments is related to data management. Since all data are volatile during the

device shell session, it was initially impossible to conduct longer experiments without the risk of losing all the captured traffic in case of connection failure. To obviate this issue, the file traces produced by the gateways (and unreliably stored on the WalT node) are automatically and periodically saved on an online repository on GitHub throughout the duration of each experiment. Moreover, when collecting data from different gateways at the same time, it was necessary to synchronize the saving operations executed from different nodes to avoid overlapping commit commands that would cause inconsistency problems. For these reasons, WalT nodes are booted with some seconds of distance one from the other and the actual beginning of the overall trace capture is considered from the second the last WalT node has been booted.

5.2 Event-driven Simulator

The simulator used for our experiments is developed in C language and implements the downlinks internal scheduling of gateways and the server behavior and logic.

It is fed with a real traffic traces file in CSV format, where each line represents a packet forwarded by a gateway to the server, as shown in Figure 5.4. The packet is the event that triggers the evolution in time of the program. Indeed, for each new line, the simulator reproduces the decision-making process of a server that receives a packet from a gateway.

Before the beginning of the virtual packets analysis, the simulator performs two operations: (a) it identifies the packets that are duplicates in the trace in order to differentiate between the number of received frames (duplicates included) and number of packets, (b) it generates the virtual downlink traffic.

It possible to perform the simulation on different network architecture, simply by specifying the number of gateways that we want in our implementation and pass it as input. Data and statistics about the packet loss are collected all along the simulation. The simulator receives also as input the range of percentage to test (e.g 0-100 means to sequentially test the network with 0, 1, 2, .., 100 % con confirmed messages), producing a file where each line refers to a specific percentage. It offers several statistical parameters, such as the percentage and the total number of confirmed message, the percentage of packets

loss because of the half-duplex mode, etc. One of the most important parameters is the Packet Delivery Rate, the states the average number of packets sent from the end-devices and effectively delivered to the server. It is calculated as follows:

$$ConfLost = ackLost_{DC} - ackLost_{SEQ} - ConfLost_{HD} \tag{5.1}$$

Confirmed packets are considered lost by the end device if its ack is lost. An ack is lost when it collides with other downlink ($\text{ackLost}_{\text{SEQ}}$ or for duty-cycle saturation ($\text{ackLost}_{\text{DC}}$). Also, a confirmed packet is lost if it is never received by the server because of the gateways half-duplex mode ($\text{ConfLost}_{\text{HD}}$).

$$LostPackets = ConfLost - UncLost_{HD} \tag{5.2}$$

The total number of lost packets is given by the sum of confirmed and the unconfirmed lost ($\text{UncLost}_{\text{HD}}$), that can just be caused by the half-duplex problem.

$$PacketsReceived = PacketsSentFromDevices - LostPackets \tag{5.3}$$

The number of the received packet is obtained subtracting the packets sent by the end-devices and the ones that are lost (and that would have been received if downlink traffic was not virtually injected in the network).

$$TotalPDR[\%] = \frac{PacketsReceived}{PacketsSentFromDevices} * 100 \tag{5.4}$$

The overall PDR percentage is computed by dividing the packets received by the packets that were originally sent by all the registered end-devices.

$$PDR_{device}[\%] = \frac{TotalPDR}{NumberOfDevices} \tag{5.5}$$

To compute the average PDR for each device, it necessary to divide for the total number of end-devices that joint the network in the experiment window.

Moreover, the simulator takes care of the gateway's status and uses it to

understand when a packet is lost or a downlink can be scheduled for a specific gateway. For each gateway, it keeps track of the number of packets in the virtual JIT queue, the ongoing transmission (if any), the sub-bands saturation, downlink requests and several statical parameters useful to determine the effective load of the gateway and its capacity to forward downlinks to the end-devices.

5.2.1 Virtual Downlink Traffic Generation

In real traffic, usually most, if not all, uplink frames are unconfirmed frames. In order to analyze the network reaction to various percentages of downlink frames, the proportion of CONF is virtually varied. In practice, the simulator randomly selects in the considered trace an unconfirmed frame (and all its duplicates) and marks it (them) confirmed. This is repeated until a new trace is produced with the targeted percentage of CONFs.

The generation of the downlink traffic, necessary to test the response of the network, is due to:

- **Confirmed Messages**. In the CSV file, the field "MODE" is always set on 'U', meaning that all the packets received are Unconfirmed. Thus, the initial file presents 0% of confirmed messages. In order to analyze the feedback of the network when exposed to different percentages of downlink load, the number of Confirmed packets is incremented. First, the number of confirmed messages corresponding to a certain percentage of the original number of packets is calculated:

$$CONFs = \frac{nb_{packets} * perc_{conf}}{100} \tag{5.6}$$

  Then the simulator randomly selects an unconfirmed packet and set its mode and that of all its duplicates (if present) to 'C' until the wanted number of confirmed messages is reached. At the end of this process a new trace is produced, where the target percentage of packets is in Confirmed mode. The simulator will then begin to scan the new list of packets. Let's suppose to test the network reaction to a downlink load of 17% on the total number packets. Notice that `nb_packet` does not correspond to the number of lines in the trace file since the

trace could (most certainly) also contain duplicates. Supposing there are no duplicates, the total number of packets is 5368 and the number of unconfirmed packets that have to be switched to confirmed mode is 912.

Since the selection of the packets to be converted is random, results can differ even if the percentage of confirmed packets is equal. To smooth this problem, the simulation is performed 60 times for each test percentage and the produced output is the average of them.

- **ADRReqBit**. All nodes are supposed to support the Adaptive Data Rate. In order to be compliant with the LoRaWAN specification, if a downlink is not received after 63 uplinks, the `ADR_REQ_BIT` is set. When this happens, the Network Server must send a downlink as soon as possible. If still after 66 uplinks, the end-devices has not received a downlink, it will switch to the next lower the data rate to try to regain connectivity with the server.

5.2.2 Packets Analysis Algorithm

When virtually receiving a new packet, the simulator will perform the following operations, as described by Algorithm 1:

1. The `ADR_ACK_COUNT` is checked. If it is equal to the `ADR_ACK_LIMIT`, the `ADR_ACK_BIT` is set for the packet.

2. Detection and collection of duplicates in the next 200 ms from the reception of the packet. A duplicate is defined as a packet sent from the same mote, with the same "Fcnt" but forwarded to the Network Server by a different gateway.

3. If the packet is an unconfirmed message, the following operations are executed:

   (a) Check if the packet is received by the server. A packet is received by the server if at least one of the duplicates reaches it. Indeed, if a gateway is sending a downlink while the packet is received, the latter is not forwarded to the Network Server and should not have to be taken into consideration. If all the duplicates are lost due to the

gateways half-duplex mode, the packet is considered definitely lost. It is evident that with more gateways, the probability of receiving duplicates increases.

(b) If `ADR_REQ_BIT` is set, try to schedule a downlink.

4. If the packet is a confirmed message, the same operations described for the unconfirmed message are performed, followed by the de-duplication process and the scheduling of the ACK.

Algorithm 1: Packets analysis algorithm

```
input : Traffic trace with confirmed packets
foreach packet that is not a duplicate do
    if ADR_ACK_LIMIT < ADR_ACK_COUNT then
        ADR_ACK_REQ_BIT = true
    end
    detect duplicates
    /* check if the analyzed packet - or at least one
       duplicate - is received by the server              */
    if hald-duplex loss and no duplicates received then
        packet lost
        update statistics
        continue
    end
    if confirmed or ADR_ACK_BIT then
        schedule a downlink
    end
end
```

5.2.3 Downlink Frames Scheduling Algorithm

In the simulation, a downlink frame can be triggered by the reception of either an unconfirmed message with the ADRACKReq bit set or a CONF that requires an ACK. If the ACK can not be scheduled, the Confirmed message is considered lost.

When a downlink has to be sent by the server, the simulator proceeds to build the downlink structure and set up all the necessary and characteristic

parameters to register the impact of the downlink transmission on the selected gateway.

- **Departure time**: It indicates the exact moment the downlink has to be sent. Its computation depends on the receive window chosen for transmission, RX1 or RX2. It is calculated as the sum of the TMPST of the internal time of the gateway's concentrator registered at the uplink arrival and 1 s (for RX1) or 2 s (for RX2).

$$DepartureTime_{RX1} = TMPST_{uplink} + 1 \tag{5.7}$$

$$DepartureTime_{RX2} = TMPST_{uplink} + 2 \tag{5.8}$$

- **Termination time**: It indicates when the downlink will be completely transmitted.

$$TerminationTime = DepartureTime + ToA_{downlink} \tag{5.9}$$

- **Sub-band Time off**: Period where the selected gateway can not send other downlink on the same sub-band.
- **SF**: The downlink is characterized by the same SF of the uplink if it is scheduled on RX1, SF12 for RX2
- **Frequency**: The downlink frequency is the same of the correspondent uplink for RX1, 869.5 for RX2
- **Size**: The downlink size is always the same and it has been chosen observing the size of real ACKs sent by the LoRaServer. The avarage is 12 bytes.

In order to send a downlink, the first operation performed by the simulator is the de-duplication process. All duplicates are analyzed and just one of them is chosen to use as the uplink of reference to build the downlink structure. Then the simulator has to select the gateway that will take care of the downlink transmission: gateways are ordered by the SNR value of the forwarded uplink frame and the "best" gateway is initially selected.

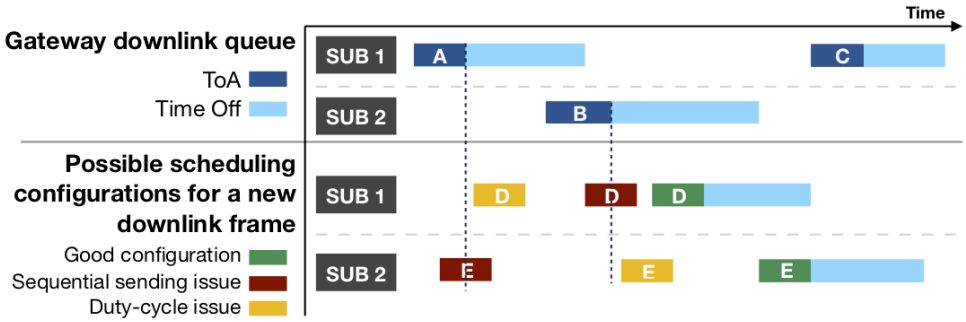


Figure 5.6: Scheduling illustration for a new downlink frame (D or E). Downlink frames A, B and C are already scheduled.

After the gateway selection, it is necessary to check three criteria able to test if the downlink can be scheduled in any of the two RX.

1. **Sequential sending**: The downlink must not overlap for all its duration with the ToA of any other downlinks already scheduled in that gateway. In Figure 5.6, this wrong configuration is given by the red packet, that overlaps with the Time on Air (in blue) of the downlink: gateways can send just a packet at a time.

2. **Duty-cycle**: The $\text{TOff}_{\text{subband}}$ must not overlap with the $\text{TOff}_{\text{subband}}$ of any other message already scheduled in that sub-band and for that gateway. The yellow packets in the figure present a wrong configuration just because of the duty cycle saturation of the sub-band 1 and 2.

3. **Downlinks collision**: If two downlinks are scheduled by different gateways at the same time, in the same channel and with the same spreading factor, a collision happens and both the downlinks are considered lost.

A first attempt is performed to schedule the downlink frame in RX1. If one of the three criteria is not met, a second attempt is performed in RX2. If again the downlink frame cannot be scheduled in RX2, then we distinguish two algorithms:

(a) **SNR-based algorithm**: the downlink frame (and thus the confirmed message associated) is considered lost and we record which unsatisfied criteria caused the loss.

(b) **Balanced algorithm**: If the gateway selection algorithm takes into consideration also the duty-cycle saturation of the sub-band, other attempts in RX1 and RX2 are performed with the second-best gateway in terms of SNR, and so on. Only when the downlink frame cannot be scheduled at any of the gateways that forwarded the corresponding uplink frame, it is considered lost.

Algorithm 2: SNR-based Gateway Selection Algorithm

```
input : Gateways list, sorted by SNR value
output: Gateway selected to transmit the downlink
maxSNR = 0
foreach gateway do
    if gateway SNR > maxSNR then
        register new best gateway
    end
end
if can be scheduled in RX1 or in RX2 then
    downlink scheduled
else
    downlink lost
end
```

Algorithm 3: Balanced Gateway Selection algorithm

```
input : Gateways list, sorted by SNR value
output: Gateway selected to transmit the downlink
while there are gateways in the list do
    maxSNR = 0
    foreach gateway do
        if gateway SNR > maxSNR then
            register new best gateway
        end
    end
    if can be scheduled in RX1 or in RX2 then
        downlink scheduled
    else
        remove gateway from the list
    end
end
if downlink not scheduled then
    downlink lost
end
```

Chapter 6

Simulation Results

This chapter is dedicated to the simulation results analysis. Different simulation experiments are performed, setting the simulator with different network architecture. The event-driven simulator produces a CSV file with raw statical information about the experiment, where each line corresponds to a percentage of downlink load. In order to produce a visually significant result, the parameters are elaborated using the R software environment.

6.1 Downlink Traffic Insights

One of the most interesting contributions of this work is to give clear insights about problems related to downlink traffic. This is done since the first step to conceive solutions is to deeply understand the origin of the issue.

In this first experiment, the simulator is fed just with the real traffic trace of the main gateway. The percentage of confirmed messages varies from 0% to 100%, in order to understand how the network will react.

It is important to underline that the packets loss here is just due to one of the causes exposed in Chapter **??**chap:3]. These packets are surely lost and would have been received if downstream traffic was not introduced in the network.

As shown in Figure 6.1, confirmed messages can be lost due to:

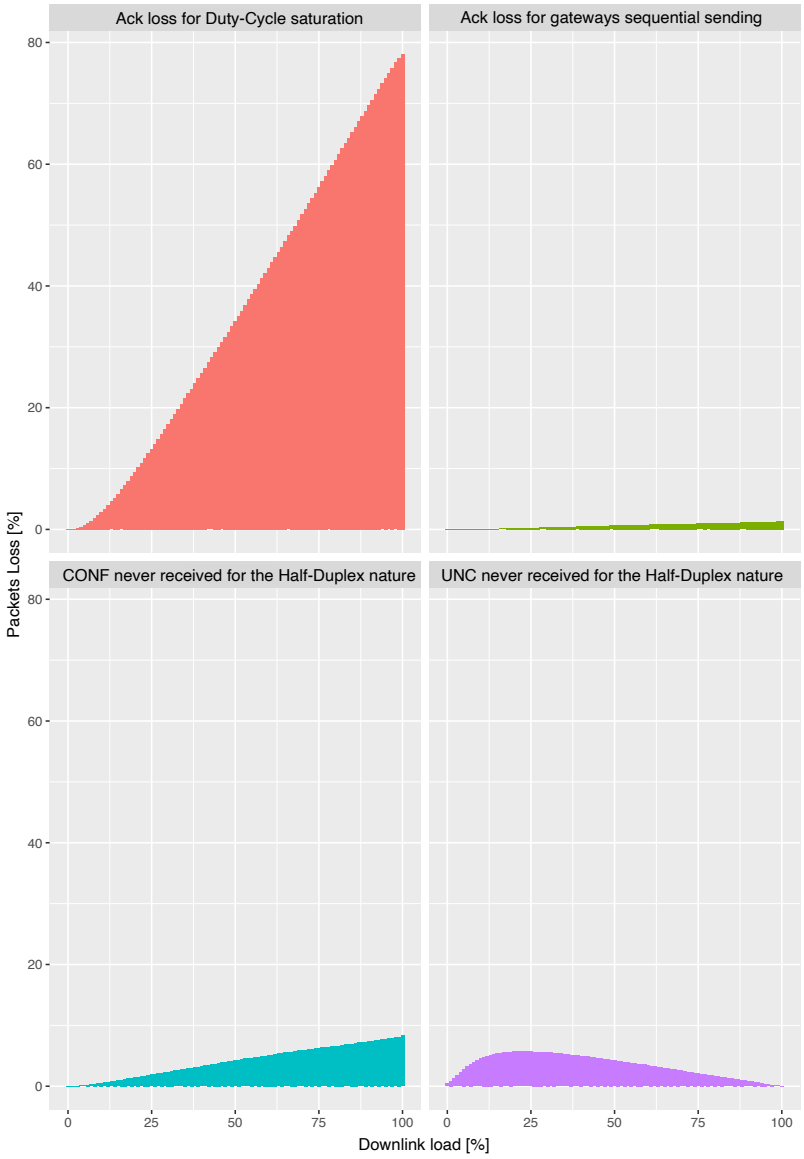


Figure 6.1: First experiment: packets loss components.

- **ACK lost**: If the scheduling of an ACK in RX1 fails, the second attempt in RX2 is performed. The ACK is lost if it can not be scheduled even in RX2. Thus, in this experiment, the unmet criterion is registered only for the failure in RX2. The loss can be caused by:
  - Failure of RX1 and duty-cycle saturation of RX2 sub-band (in red)
  - Failure of RX1 and gateway's downlinks sequential sending for RX2 start time (in green)
- **Half-duplex mode** of the gateway (in cyan): the confirmed message was

supposed to be received while the gateway is sending another downlink and, for this reason, it is lost.

Differently, unconfirmed packets can be lost just because of the gateway's half-duplex mode (in purple).

The sum of all the four components gives the total packets loss over the percentage of confirmed messages.

Figure 6.2 highlights three main factors:

1. With a low percentage of CONF, most losses are related to unconfirmed messages. Indeed, if the downlink load is low, it does not congest the gateway and ACKs can be sent. On the other hand, every time a downlink frame is sent, all the uplink frames received during that period are lost.

2. From 20% of CONF and on, it is possible to notice how the main cause of frame loss is duty-cycle saturation. If all the frames are confirmed (100% of CONF), 8% of them are not received by the gateway because of its half-duplex nature, and 76% of the requested ACKs cannot be transmitted. In this case, the frame loss reaches 86%: almost all frames are lost.

3. The sequential sending of downlink frames causes no more than 2% of losses even with 100% CONF. This can be explained by observing that a heavier traffic load than the one deployed here would correspond to a higher downlink frames demand. With more downlink frames requested, the probability of overlapping with other downlink frames already scheduled in the gateway would surely increase. Since the main ACK loss is due to the duty-cycle saturation, the parallel sending solution will not produce a tangible improvement with these data (its validation is left for future work).

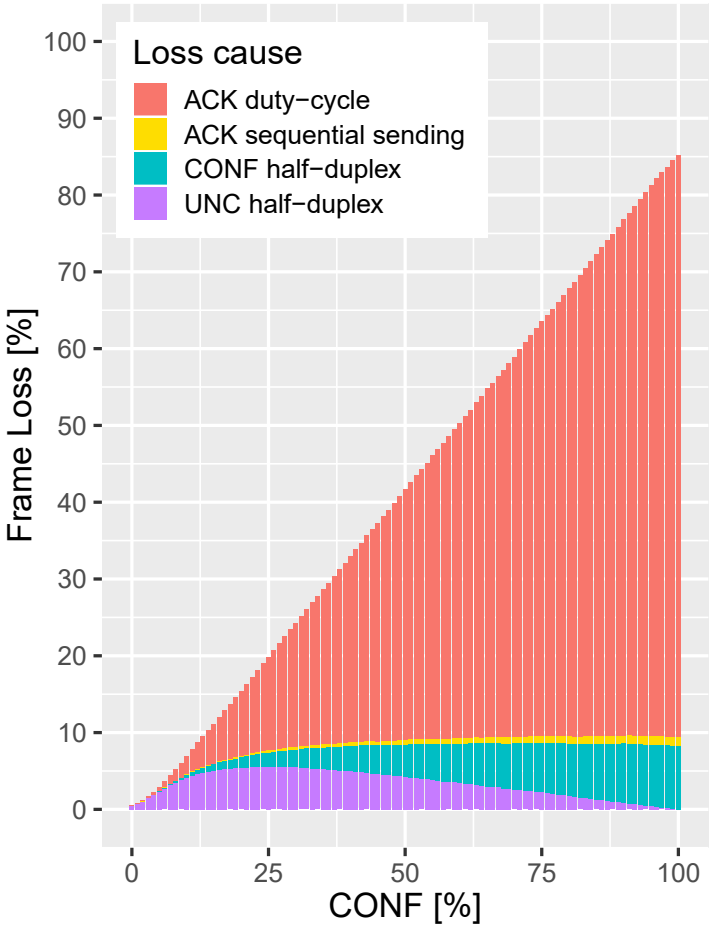


Figure 6.2: First experiment: packet loss due to downlink traffic with a single-gateway implementation.

6.2 Multi-gateway Architecture

A second simulation is performed in order to test the improvement brought by a multi-gateway implementation.

The simulator is fed with a real traffic trace file that combines the packets received by the main gateway and the duplicates received, in the same time window, by the other three gateways: the first combination tests a two-gateways implementation, where the second gateway is GW2. The second combination adds to the first one also the duplicates received by GW3, and the fourth adds to the third the duplicates of GW4, finally having a quad-gateway implementation.

This solution aims at decreasing the packets loss due to the gateways half-

duplex mode, as shown in Figure 6.3.

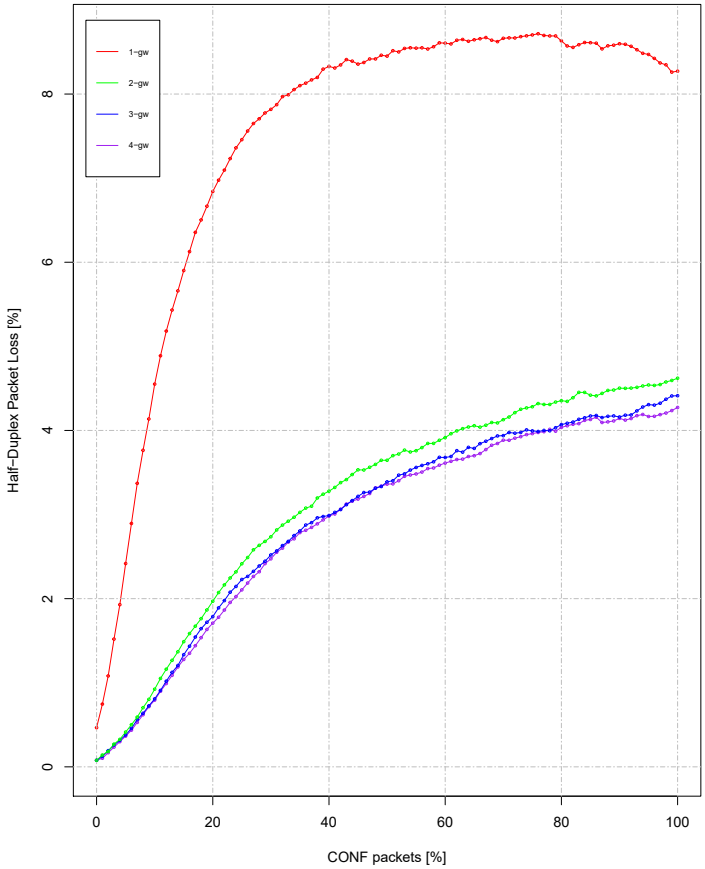


Figure 6.3: Second experiment. Packet loss due to the half-duplex mode deploying 1 up to 4 gateways.

As expected, the packet loss sensitively decreases with the increasing number of gateway deployed in the architecture, and with a quad-gateway implementation the loss, due to the half-duplex nature of LoRa gateways, is almost halved. We start with a loss just over 8% in a single gateway network exposed to a downlink load of 100%, and progressively move to loss of almost 4% in a quad-gateway implementation under the same load. Indeed, all the packets that are lost deploying just one gateway are received by the other *free* gateways. The big improvement brought by the addition of the second gateway is due to the fact that it receives almost all the traffic also received by the main one.

It is also interesting to observe the total packets loss insights shown in Figure 6.4: in a single-gateway implementation, all the downlinks are requested to be sent through the same gateway, whereas, in a multi-gateway scenario, the server has to select the gateway that will transmit the downlink. Having more available gateways can help to spread the downstream traffic through more base stations and thus, decrease the congestion.

As anticipated, also the general packet loss is decreased. In particular, it is possible to notice a sensitive reduction of losses due to duty-cycle saturation. The downlink traffic load per gateway is lower, more downlinks can be sent in RX1 and the RX2 sub-band is slowly saturated.

It is worth noting that in this scenario, the server chooses the gateway that will forward the downlink just based on the best SNR value shown by the duplicates.

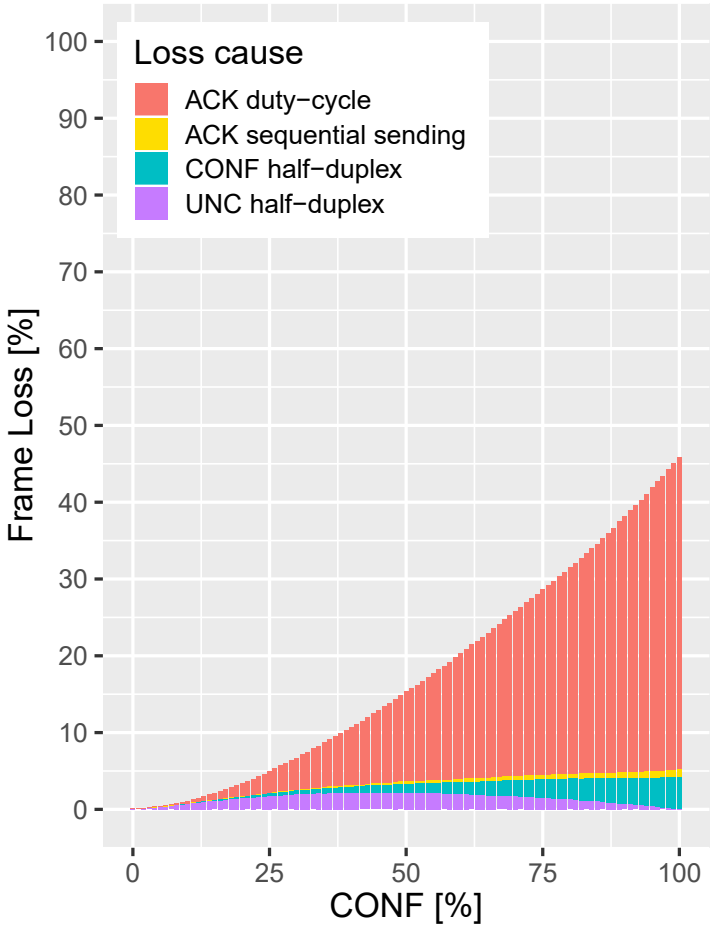


Figure 6.4: Second experiment. Packet loss due to downstream traffic with a quad-gateway implementation.

6.3 Gateway Selection Algorithms Comparison

One of the main keys to improving the downlink scalability in the network is an accurate study of the best gateway selection procedure each time a downlink has to be scheduled for transmission. The best gateway selection algorithm should always guarantee an acceptable Packet Delivery Rate even in case of a high confirmed message load. In this case, the PDR is strictly correlated with the gateway's capacity to carry on the ACK transmission, since a CONF with a missed ACK is considered lost by the end-device. The aim of the following experiments is to present and evaluate two different selection algorithms:

first, the SNR-based selection algorithm is analyzed, then then the Balanced selection algorithm is tested on the same traffic trace and the two are finally compared.

6.3.1 SNR-based Selection Algorithm

As shown in Section 6.2, the deployment of more gateways helps to reduce packet loss especially when the CONF packets percentage is low. Here, despite the use of more gateways, frame loss for a higher proportion of confirmed messages is still important (around 40%). This is a direct consequence of the naive behavior of the SNR-based gateway selection algorithm which does not attempt to spread downlink traffic in any way by always and only selecting the gateway with the best reception conditions. The SNR-based algorithm produces a more balanced downlink traffic load just if the best gateway in terms of SNR is not always the same.

In order to observe the disproportionate downlink load on the gateways caused by the algorithm, the simulator is fed with the traffic traces of the four gateways and the gateways load is analyzed. Here, this concentration of downlink traffic is clear in Figure 6.5, which analyzes the behavior of the gateways under the SNR-based selection algorithm. Each time a gateway is selected, the simulator increment its downlink request counter. Comparing the counters of the gateways it possible to observe if one of them is favored with respect to the others. Figure 6.5a shows that GW1 is selected at least 60% of the time when an ACK has to be scheduled, GW2 (despite being locally displaced very closed to GW1) is chosen from 40 up to 30 % of the time.

To understand how this imbalance impacts the network capacity, the Ack Sending Rate (ASR) is computed. As the PDR indicates the percentage of packets that are effectively delivered, the ASR wants to provide the percentage of ACK effectively sent. It is computed as follow:

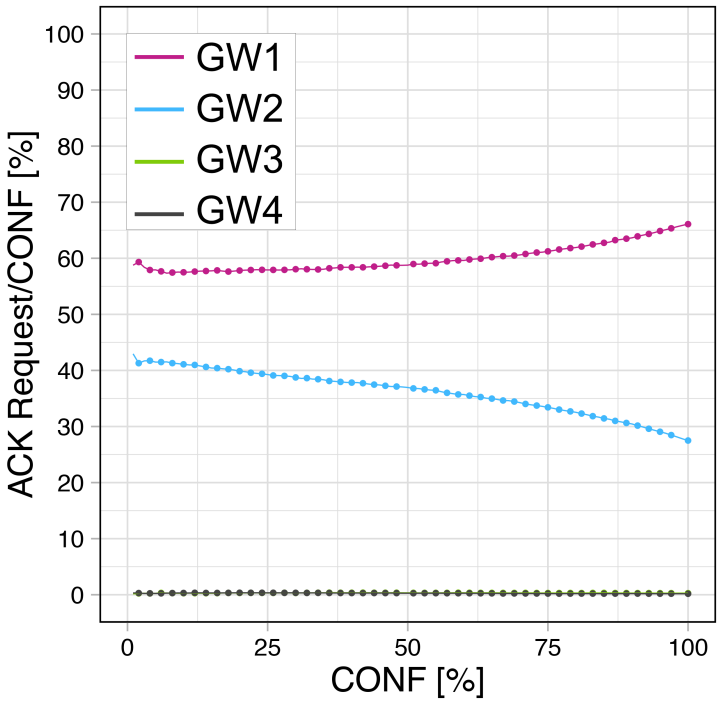
$$ASR = \frac{ACK_{sent}}{ACK_{requested}} \tag{6.1}$$

A low ASR value states the gateway inability to forwards downlinks and the indication of possible bottlenecks in the network. Indeed, when the CONF percentage is high, the gateway ability to forwards the ACK is vital to avoid

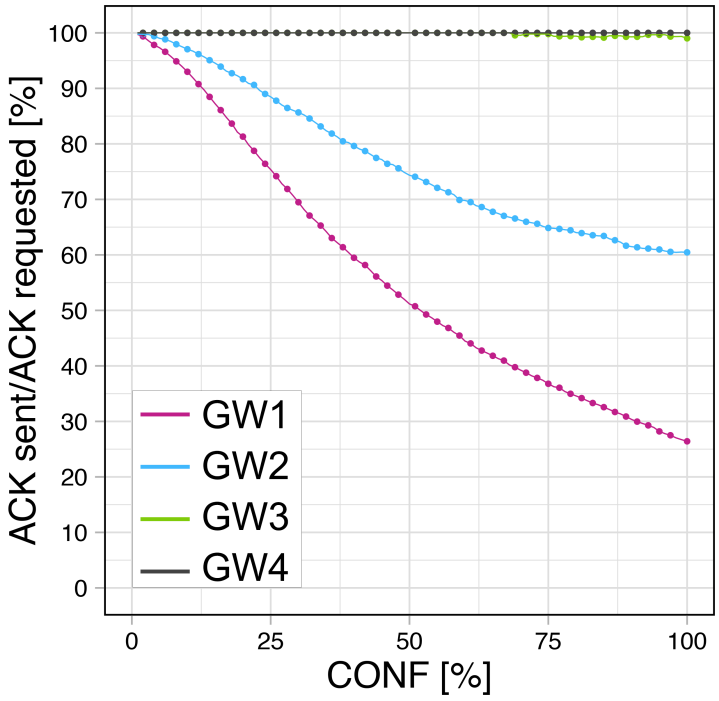
total network congestion and the decrease of the PDR for each end-device.

In Figure 6.5b it is possible to observe that, as a consequence of such high downlink frame request load, the effective percentage of ACKs sent by the gateway significantly decreases with the increasing load. With 100% of CONF, GW1 is only able to forward 24% of the requested ACKs. On the other side, two gateways (GW3 and GW4) are almost never used, while GW2 is complementary to GW1.

In this simulation, it is clear that the SNR-based gateway selection algorithm can be a good choice just for network architectures that are lightly loaded with downlink requests or that have a fixed and well-studied configuration, where the radio conditions of nodes does not vary in time, and their displacement with respect to the gateways can guarantee an optimal distribution of downlink load even if the selection is just based on the SNR value. On the other hand, it does not provide a scalable solution for large dimensions networks where radio conditions can change. The gateways quickly overload and lose their ability to forwards packets, saturating the different sub-bands duty-cycle.



(a) Percentage of ACK requested per gateway under total number of CONF frames.



(b) Percentage of ACK sent on ACK requested per gateway.

Figure 6.5: Gateways behavior under SNR-based selection algorithm.

6.3.2 Balanced Selection Algorithm

To better spread the load of the downlink traffic, the balanced gateway selection algorithm is introduced. The latter is tested on the same traffic trace of the preceding experiment, to produce comparable results.

The main modification introduced by this algorithm is the possibility to keep track of the gateways load an, as a consequence, to iterate the gateway selection if the downlink is predicted to be lost in the selected gateway conditions. It is important to highlight that storing information about the gateway is not a modification that requires additional hardware, but it is a solution immediately deployable within the current LoRaWAN architectures. Indeed, it is just sufficient to modify the Network Server software to let it check, when needed, the gateway load.

To compare this algorithm with the SNR-based one and to show at best the improvements it introduces, the frame loss is chosen as the main parameter.

The improvements introduced by the balanced algorithm are shown in Figure 6.6. It is observable how, in a quad-gateway implementation (in green), the balanced algorithm decreases the frame loss by 25% compared to the SNR-based algorithm and by 66% compared to the single-gateway architecture (in red), producing losses that never exceed 20% of the total traffic. The main reason for this improvement is the better distribution of downlink traffic. ACK loss, which was the main cause of frame loss for the high percentage of CONF, is strongly decreased, as clearly shown in Figure 6.7. The percentage of ACK loss relative to the number of CONF percentage tested tends to increase with the CONF load in the network. Nevertheless, Figure 6.7b shows how the ACK loss percentage for the Balanced selection algorithm is halved. Unlike before, a downlink frame is not scheduled on the gateway with the best SNR if it is duty-cycle saturated for the given sub-bands or it overlaps with other already scheduled downlink frames. The algorithm tries the second-best gateway in terms of SNR, and so on. Obviously, ACKs can still be lost due to other factors, especially if transmitted by a gateway that presents a lower SNR value. But in fact, the downlink channel is less challenging than the uplink as in all cellular networks: nodes are in view of less contending entities than the gateways, which use a specific modulation and are generally placed so as to receive from as many nodes as possible.

Nevertheless, it is worth highlighting the ability of LoRa devices to receive signal powers below the receiver noise floor. Indeed, the performance of the LoRa modulation itself, forward error correction (FEC) techniques and the spread spectrum processing gain combine to allow significant SNR improvements. Most of the LoRa packets received in the captured traces are below the noise floor and correctly demodulated by the gateway.

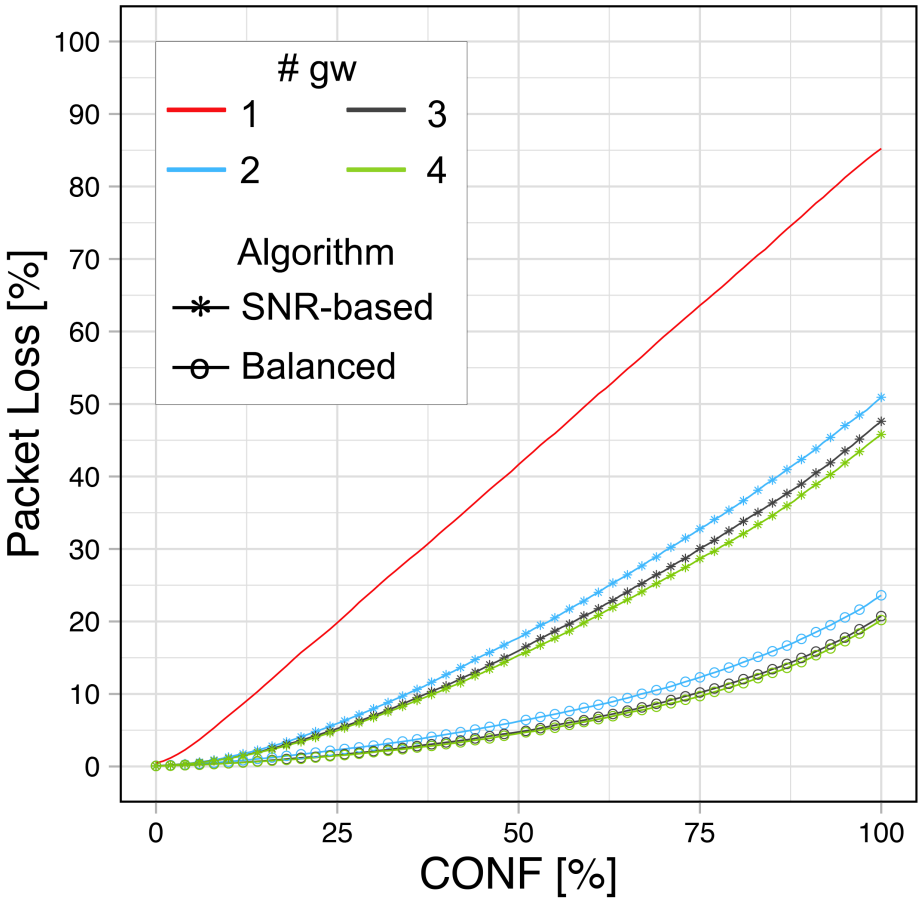
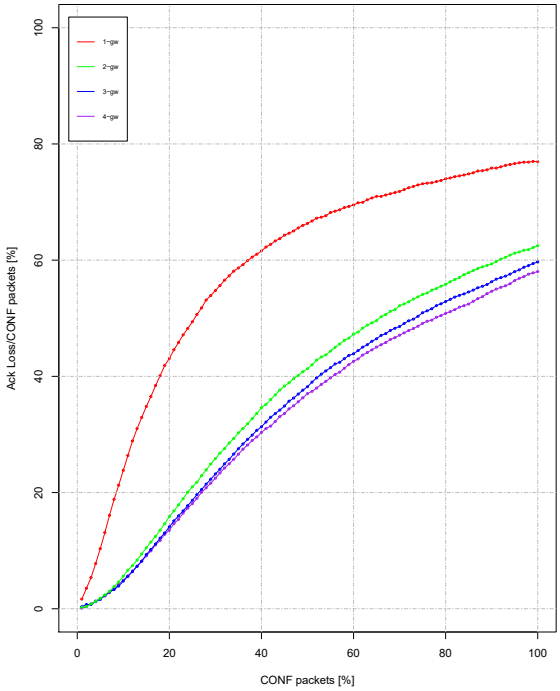
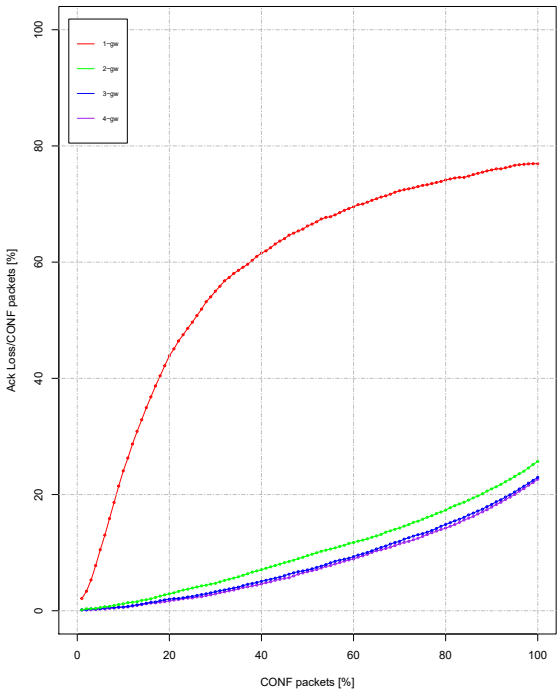


Figure 6.6: Frame loss percentage under different algorithms and number of deployed gateways.



(a) ACK loss with one to four gateways architecture and SNR-based algorithm



(b) ACK loss with one to four gateways architecture and Balanced selection algorithm.

Figure 6.7: Percentage of ACK loss on the total number of CONF packets, at the variation of CONF packets percentage in the network and with different selection algorithms.

Chapter 7

Conclusion

7.1 Summary

In this work, a comprehensive analysis of the impact of downlink traffic on the network capacity is presented. In a first phase, the roots of the problem are identified: the half-duplex mode of LoRa gateways impedes the reception of uplinks when transmitting a downlink; the duty-cycle causes the saturation of the downlink transmission sub-bands making the gateways the bottleneck of the network; and the downlink sequential scheduling exacerbates the above-cited problematics by forbidding parallel downlink transmissions. These three problematics are identified as the origin of the well-known increase of packet loss that occurs when downlink traffic is introduced in the network.

Three solutions have been proposed in order to mitigate such harmful effects:

1. A multi-gateway implementation allows the reception of uplinks by the other free gateways in the network.

2. Two gateway selection algorithms are studied and evaluated: the SNR-based selection algorithm and a balanced gateway selection algorithm. The latter guarantees a correct downlink traffic spread among the different gateways, preventing their premature saturation.

3. The downlinks parallel sending shorten the no-reception-window of the gateways.

With the purpose of inspecting the identified problems and validating the proposed solutions, an event-driven simulator fed with real-world traffic traces has been developed. The simulator allows the testing of different network architectures, by combining as many traffic trace captures as desired number of gateways. The network is virtually exposed to various percentages of downlink load.

The simulation results show that the main cause of packet loss is the duty-cycle saturation, and that, thanks to the deployment of more gateways and a more balanced downlink load, such negative impact can be alleviated.

The deployment of these solutions enable the successful realization of a higher percentage of LoRaWAN confirmed messages. The engineering of IoT applications that require such confirmed data communication can be facilitated. Besides, the deprecation of confirmed message for low and medium downlink traffic load could be not necessary anymore.

7.2 Future Work

As future work, it would be surely interesting to evaluate the correlation between the SNR values and the prospect of downlink reception, in order to understand the real probability for downlinks to be correctly delivered when sent through a gateway that presented a low SNR for the correspondent uplink frame. In this context, it is important to highlight how LoRa offers a vast number of parameters that can be characterized with the aim of finding the perfect combination to achieve the highest possible SNR for a given device. Indeed, even just modifying the spreading factor or increasing the transmission power (if the end device battery allows it), would improve the SNR value.

Other radio parameters can be chosen to perform the gateway selection algorithm, such as the Received Signal Strength Indicator (RSSI). It could also be interesting the evaluation of a selection algorithm that completely ignores any radio parameters and just takes into account the downlink load per gateway. The only reason to apply this algorithm is to fasten the selection procedure, to the detriment of more reliable transmissions.

Moreover, performing the experiments within a larger network could provide better insights and a more realistic quantification of the improvements.

Besides, in such a scenario, it would be reasonable to implement the downlinks parallel sending, also supposing a bigger packet loss exclusively caused by it. Experimental set up to validate the feasibility of the parallel sending could also be realized by deploying two gateways behaving like one and sending downlinks at the same time.

Another compelling work would be to combine the proposed solutions with other mechanisms able to enhance the communication reliability, in order to further decrease the packets loss even in scenarios with high load of confirmed messages.

Bibliography

Literary Sources

[1] J. M. Marais, A. M. Abu-Mahfouz, and G. P. Hancke, "A survey on the viability of confirmed traffic in a lorawan", *IEEE Access*, vol. 8, pp. 9296–9311, 2020, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2964909.

[2] S. Böcker, C. Arendt, P. Jörke, and C. Wietfeld, "Lpwan in the context of 5g: Capability of lorawan to contribute to mmtc", in *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, 2019, pp. 737–742. DOI: 10.1109/WF-IoT.2019.8767333.

[3] J. Navarro-Ortiz, J. Ramos, J. Lopez-Soler, C. Cervelló-Pastor, and M. Catalan, "A lorawan testbed design for supporting critical situations: Prototype and evaluation", *Wireless Communications and Mobile Computing*, vol. 2019, pp. 1–12, Feb. 2019. DOI: 10.1155/2019/1684906.

[4] D. Poluektov, M. Polovov, P. Kharin, M. Štůsek, K. Zeman, P. Masek, I. Gudkova, J. Hosek, and K. Samouylov, "On the performance of lorawan in smart city: End-device design and communication coverage", in. Dec. 2019, pp. 15–29, ISBN: 978-3-030-36613-1. DOI: 10.1007/978-3-030-36614-8_2.

[5] W. Saadeh, S. A. Butt, and M. A. B. Altaf, "A patient-specific single sensor iot-based wearable fall prediction and detection system", *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 27, no. 5, pp. 995–1003, 2019, ISSN: 1558-0210. DOI: 10.1109/TNSRE.2019.2911602.

[6] E. A.A.A. B. T., "A survey on lorawan architecture, protocol and technologies.", in *Future Internet*, vol. 11, 2019, p. 216. DOI: 10.3390/fi11100216.

[7] M. Capuzzo, D. Magrin, and A. Zanella, "Confirmed traffic in lorawan: Pitfalls and countermeasures", in *2018 17th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, 2018, pp. 1–7.

[8] 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*, vol. 6, pp. 1–13, 2018, ISSN: 2168-2372. DOI: 10.1109/JTEHM.2018.2822302.

[9] M. Centenaro and L. Vangelista, "Boosting network capacity in LoRaWAN through time-power multiplexing", in *Proc. IEEE PIMRC*, Bologna, Italy, 2018, pp. 1–6.

[10] H. Chaudhary, B. Tank, and H. Patel, "Comparative analysis of internet of things (iot) based low power wireless technologies", vol. V7, Jan. 2018.

[11] L. Feltrin, C. Buratti, E. Vinciarelli, R. De Bonis, and R. Verdone, "Lorawan: Evaluation of link- and system-level performance", *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2249–2258, 2018, ISSN: 2372-2541. DOI: 10.1109/JIOT.2018.2828867.

[12] G. Ferré and A. Giremus, "Lora physical layer principle and performance analysis", in *2018 25th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2018, pp. 65–68. DOI: 10.1109/ICECS.2018.8617880.

[13] Q. Lone, E. Dublé, F. Rousseau, I. Moerman, S. Giannoulis, and A. Duda, "Wish-walt: A framework for controllable and reproducible lora testbeds", in *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2018, pp. 1–7. DOI: 10.1109/PIMRC.2018.8580722.

[14] LoRa Alliance Technical Committee Regional Parameters Workgroup, "LoRaWAN™ 1.1 specification", LoRa Alliance, Tech. Rep. 2018.

[15] L. V. Michele Luvisotto Federico Tramarin and S. Vitturi, "On the use of lorawan for indoor industrial iot applications", *Wireless Communications and Mobile Computing*, vol. 2018, p. 11, 2018. DOI: https://doi.org/10.1155/2018/3982646.

[16] K. Mikhaylov, J. Petäjäjärvi, and A. Pouttu, "Effect of downlink traffic on performance of lorawan lpwa networks: Empirical study", in *2018 IEEE 29th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 2018.

[17] T. Miriyala, R. Karthik, J Mahitha, and V Reddy, "Iot based forest fire detection system", *International Journal of Engineering and Technology*, vol. 7, p. 124, Mar. 2018. DOI: 10.14419/ijet.v7i2.7.10277.

[18] Semtech Corporation, "Sx1257 low power digital I and Q RF multi-PHY mode transceiver", Rev. 1.2, 2018.

[19] C. Yoon, M. Huh, S.-G. Kang, J. Park, and C. Lee, "Implement smart farm with iot technology", in *2018 20th International Conference on Advanced Communication Technology (ICACT)*, IEEE, 2018, pp. 749–752.

[20] F. V. den Abeele, J. Haxhibeqiri, I. Moerman, and J. Hoebeke, "Scalability analysis of large-scale LoRaWAN networks in ns-3", *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 2186–2198, 2017. DOI: 10.1109/JIOT.2017.2768498.

[21] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, "Understanding the limits of LoRaWAN", *IEEE Communications Magazine*, vol. 55, no. 9, pp. 34–40, 2017, ISSN: 0163-6804. DOI: 10.1109/MCOM.2017.1600613.

[22] S. B. Baker, W. Xiang, and I. Atkinson, "Internet of things for smart healthcare: Technologies, challenges, and opportunities", *IEEE Access*, vol. 5, pp. 26 521–26 544, 2017, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2017.2775180.

[23] D. Bankov, E. Khorov, and A. Lyakhov, "Mathematical model of lorawan channel access", in *2017 IEEE 18th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2017, pp. 1–3. DOI: 10.1109/WoWMoM.2017.7974300.

[24] D. Bankov, E. Khorov, and A Lyakhov, "Mathematical model of lorawan channel access", Jun. 2017.

[25] M. Centenaro, L. Vangelista, and R. Kohno, "On the impact of downlink feedback on LoRa performance", in *IEEE Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 2017, pp. 1–6. DOI: 10.1109/PIMRC.2017.8292315.

[26] A. Gemalto and Semtech, "Lorawan security whitepaper", LoRa Alliance, Tech. Rep. 2017.

[27] O. Georgiou and U. Raza, "Low power wide area network analysis: Can lora scale?", *IEEE Wireless Communications Letters*, vol. 6, no. 2, pp. 162–165, 2017, ISSN: 2162-2337. DOI: 10.1109/LWC.2016.2647247.

[29] W. Guibene, J. Nowack, N. Chalikias, K. Fitzgibbon, M. Kelly, and D. Prendergast, "Evaluation of lpwan technologies for smart cities: River monitoring use-case", in *2017 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, IEEE, 2017, pp. 1–5.

[30] V. Gupta, S. K. Devar, N. H. Kumar, and K. P. Bagadi, "Modelling of iot traffic and its impact on lorawan", in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, 2017, pp. 1–6. DOI: 10.1109/GLOCOM.2017.8254512.

[31] U. Noreen, A. Bounceur, and L. Clavier, "A study of lora low power and wide area network technology", in *2017 International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)*, 2017, pp. 1–6. DOI: 10.1109/ATSIP.2017.8075570.

[32] A. Pop, U. Raza, P. Kulkarni, and M. Sooriyabandara, "Does bidirectional traffic do more harm than good in LoRaWAN based LPWA networks?", in *2017 IEEE Global Communications Conference, GLOBECOM 2017, Singapore, December 4-8, 2017*, 2017, pp. 1–6. DOI: 10.1109/GLOCOM.2017.8254509. [Online]. Available: https://doi.org/10.1109/GLOCOM.2017.8254509.

[33] A. Pötsch and F. Haslhofer, "Practical limitations for deployment of lora gateways", in *2017 IEEE International Workshop on Measurement and Networking (M N)*, 2017, pp. 1–6. DOI: 10.1109/IWMN.2017.8078360.

[35] N. Sornin and A. Yegin, "LoRaWAN™ 1.1 specification", LoRa Alliance, Tech. Rep. 2017.

[36] T. Truong, A. Dinh, and K. Wahid, "An iot environmental data collection system for fungal detection in crop fields", 2017, pp. 1–4. DOI: 10.1109/CCECE.2017.7946787.

[37] F. Yu, Z. Zhu, and Z. Fan, "Study on the feasibility of lorawan for smart city applications", in *2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2017, pp. 334–340. DOI: 10.1109/WiMOB.2017.8115748.

[38] Adeunis, "Field test device lorawan europe eu863-8701.1 specification", Adeunis, Tech. Rep. 2016.

[39] A. Augustin, J. Yi, T. Clausen, and W. M. Townsley, "A study of lora: Long range and low power networks for the internet of things", *Sensors*, vol. 16, no. 9, 2016.

[40] J. Bardyn, T. Melly, O. Seller, and N. Sornin, "Iot: The era of lpwan is starting now", in *ESSCIRC Conference 2016: 42nd European Solid-State Circuits Conference*, 2016, pp. 25–30. DOI: 10.1109/ESSCIRC.2016.7598235.

[42] M. Bor, U. Roedig, T. Voigt, and J. M Alonso, "Do lora low-power wide-area networks scale?", Nov. 2016.

[43] P. Brunisholz, E. Duble, F. Rousseau, and A. Duda, "Walt: A reproducible testbed for reproducible network experiments", in *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2016, pp. 146–151. DOI: 10.1109/INFCOMW.2016.7562062.

[44] M. Centenaro, L. Vangelista, A. Zanella, and M. Zorzi, "Long-range communications in unlicensed bands: The rising stars in the iot and smart city scenarios", *IEEE Wireless Communications*, vol. 23, no. 5, pp. 60–67, 2016, ISSN: 1536-1284. DOI: 10.1109/MWC.2016.7721743.

[45] M. Elkhodr, S. Shahrestani, and H. Cheung, "Emerging wireless technologies in the internet of things : A comparative study", *International Journal of Wireless and Mobile Networks*, vol. 8, Nov. 2016. DOI: 10.5121/ijwmn.2016.8505.

[47] F. Orfei, C. Benedetta Mezzetti, and F. Cottone, "Vibrations powered lora sensor: An electromechanical energy harvester working on a real bridge", in *2016 IEEE SENSORS*, 2016, pp. 1–3. DOI: 10.1109/ICSENS.2016.7808752.

[48] M. R. Palattella, M. Dohler, A. Grieco, G. Rizzo, J. Torsner, T. Engel, and L. Ladid, "Internet of things in the 5g era: Enablers, architecture, and business models", *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 3, pp. 510–527, 2016, ISSN: 1558-0008. DOI: 10.1109/JSAC.2016.2525418.

[49] 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*, Oct. 2015. DOI: 10.4108/eai.26-10-2015.150597. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01231221.

[50] Semtech Corporation, *LoRa modulation basics*, www.semtech.com, 2015.

[51] L. Vangelista, A. Zanella, and M. Zorzi, "Long-range iot technologies: The dawn of lora™", in *Future Access Enablers for Ubiquitous and Intelligent Infrastructures*, V. Atanasovski and A. Leon-Garcia, Eds., Cham: Springer International Publishing, 2015, pp. 51–58, ISBN: 978-3-319-27072-2.

[52] F. Boccardi, R. W. Heath, A. Lozano, T. L. Marzetta, and P. Popovski, "Five disruptive technology directions for 5g", *IEEE Communications Magazine*, vol. 52, no. 2, pp. 74–80, 2014, ISSN: 0163-6804. DOI: 10.1109/MCOM.2014.6736746.

[53] M.Research, "The need for low cost,high reach,wide area connectivity for the internet of things. a mobile network operator's perspective.", white paper, 2014.

[54] European Telecommunications Standards Institute, "Electromagnetic compatibility and radio spectrum matters (ERM); short range devices (SRD); radio equipment in the 25 MHz to 1000 MHz with power levels up to 500 mW", European Telecommunications Standards Institute, Tech. Rep. EN300 4.1, 2013, pp. 220–1.

[55] N. Maalel, E. Natalizio, A. Bouabdallah, P. Roux, and M. Kellil, "Reliability for emergency applications in internet of things", in *2013 IEEE International Conference on Distributed Computing in Sensor Systems*, 2013, pp. 361–366. DOI: 10.1109/DCOSS.2013.40.

[56] N. Maalel, E. Natalizio, A. Bouabdallah, P. Roux, and M. Kellil, "Reliability for emergency applications in internet of things", in *2013 IEEE International Conference on Distributed Computing in Sensor Systems*, 2013, pp. 361–366. DOI: 10.1109/DCOSS.2013.40.

[57] Semtech Corporation, *Sx1272/3/6/7/8 LoRa modem design guide*, 2013.

[58] L. Yang, S.-H. Yang, and L. Plotnick, "How the internet of things technology enhances emergency response operations. technological forecasting and social change, 80(9), 1854-1867", vol. 80, pp. 1854–1867, Nov. 2013.

[59] R. Khan, S. U. Khan, R. Zaheer, and S. Khan, "Future internet: The internet of things architecture, possible applications and key challenges", in *2012 10th International Conference on Frontiers of Information Technology*, 2012, pp. 257–260. DOI: 10.1109/FIT.2012.53.

[60] W. L. Tan, P. Hu, and M. Portmann, "Snr-based link quality estimation", in *2012 IEEE 75th Vehicular Technology Conference (VTC Spring)*, 2012, pp. 1–5. DOI: 10.1109/VETECS.2012.6239865.

[61] D. Bandyopadhyay and J. Sen, "Internet of things: Applications and challenges in technology and standardization", *Wireless Personal Communications*, vol. 58, pp. 49 –69, May 2011. DOI: 10.1007/s11277-011-0288-5.

[62] S. Lien, K. Chen, and Y. Lin, "Toward ubiquitous massive accesses in 3gpp machine-to-machine communications", *IEEE Communications Magazine*, vol. 49, no. 4, pp. 66–74, 2011, ISSN: 0163-6804. DOI: 10.1109/MCOM.2011.5741148.

[63] A. Springer, W. Gugler, M. Huemer, L. Reindl, C. C. W. Ruppel, and R. Weigel, "Spread spectrum communications using chirp signals", in *IEEE/AFCEA EUROCOMM 2000. Information Systems for Enhanced Public Safety and Security (Cat. No.00EX405)*, 2000, pp. 166–170. DOI: 10.1109/EURCOM.2000.874794.

[64] Semtech Corporation, *LoRa PicoCell gateway user guide*, www.semtech.com.

Web Sources

[28] S. Ghoslya. (2017). Matlab code to generate lora symbols. http://www.sghoslya.com/p/in-previous-post-you-read-theory-about.html.

[34] Semtech. (2017). Lora network packet forwarder project. https://github.com/Lora-net/packet_forwarder/blob/master/lora_pkt_fwd.

[41] J. Blum. (2016). Lora modem with limesdr. https://myriadrf.org/news/lora-modem-limesdr/.

[46] A. Nordrum. (2016). Popular internet of things forecast of 50 billion devices by 2020 is outdated. https://spectrum.ieee.org/tech-talk/telecom/internet/popular-internet-of-things-forecast-of-50-billion-devices-by-2020-is-outdated.