UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

NETWORK SIZE ESTIMATION FOR LORA-BASED DIRECT-TO-SATELLITE IOT

TESIS PARA OPTAR AL GRADO DE
MAGÍSTER EN CIENCIAS, MENCIÓN COMPUTACIÓN

DIEGO ANTONIO MALDONADO MUÑOZ

PROFESORES GUÍA:
JAVIER BUSTOS JIMÉNEZ
SANDRA CÉSPEDES UMAÑA

PROFESOR CO-GUÍA
JUAN FRAIRE

MIEMBROS DE LA COMISIÓN:
RODRIGO ARENAS ANDRADE
IVANA BACHMANN ESPINOZA
CRISTIAN RUZ RUZ

Esta tesis ha sido financiada parcialmente por
ANID Fondecyt Regular 1201893 y ANID Basal Project FB0008.

SANTIAGO DE CHILE
2023

RESUMEN DE LA TESIS PARA OPTAR AL GRADO DE
MAGÍSTER EN CIENCIAS, MENCIÓN COMPUTACIÓN
POR: DIEGO ANTONIO MALDONADO MUÑOZ
FECHA: 2023
PROF. GUÍA: SANDRA CÉSPEDES & JAVIER BUSTOS

ESTIMACIÓN DEL TAMAÑO DE RED PARA IOT DIRECTO A SATÉLITE BASADO EN LORA

El paradigma emergente de la Internet de las Cosas Directo a Satélite (DtS-IoT) implica que los nodos de la superficie terrestre se comuniquen directamente con satélites de órbita terrestre baja (LEO), utilizando protocolos estándar de redes de área amplia de baja potencia (LPWAN).

Uno de los principales retos a los que se enfrenta este paradigma es la ampliación del control de acceso al medio (MAC) de un número limitado de nodos a miles dentro de la zona de cobertura del satélite. Para resolver este problema, los esquemas de control de acceso al medio pueden utilizar información a priori sobre el número de nodos que el satélite cubrirá a lo largo de su órbita. Sin embargo, el desarrollo de soluciones técnicamente viables para la estimación del tamaño de la red que sean a la vez precisas y exactas sigue siendo un reto de investigación abierto. Este trabajo presenta la implementación, selección de parámetros y evaluación del primer protocolo de estimación del tamaño de red compatible con LoRa/LoRaWAN que aprovecha el estimador de información de colisión optimista (OCI) de a bordo.

Nuestra solución, LoRa-OCI (L-OCI), se integró en FLoRaSat, un simulador C++ de eventos discretos DtS-IoT que integra modelos orbitales y de comunicación LoRa/LoRaWAN realistas. A través de una amplia campaña de simulación, podemos determinar las configuraciones LoRa adecuadas para lograr un error cuadrático medio (RMSE) bajo y un bajo consumo de energía. Además, nuestros resultados indican que el enfoque es relativamente insensible a los parámetros LoRa cuando se evalúa el rendimiento agregado de un protocolo Slotted ALOHA Game (SAG) provisto de estimaciones de L-OCI.

RESUMEN DE LA TESIS PARA OPTAR AL GRADO DE
MAGÍSTER EN CIENCIAS, MENCIÓN COMPUTACIÓN
POR: DIEGO ANTONIO MALDONADO MUÑOZ
FECHA: 2023
PROF. GUÍA: SANDRA CÉSPEDES & JAVIER BUSTOS

NETWORK SIZE ESTIMATION FOR LORA-BASED DIRECT-TO-SATELLITE IOT

The emerging paradigm of Direct-to-Satellite Internet of Things (DtS-IoT) involves Earth surface nodes communicating directly with Low Earth Orbit (LEO) satellites, utilizing standard Low-Power Wide Area Networks (LPWAN) protocols.

One of the core challenges faced in this paradigm is scaling the Medium Access Control (MAC) from a limited number of nodes to potentially thousands within the satellite's coverage area. To address this issue, medium access control schemes can utilize a priori information on the number of nodes the satellite will cover along its orbit. However, developing technically viable solutions for network size estimation that are both precise and accurate remains an open research challenge. This work presents the implementation, parameter selection, and evaluation of the first LoRa/LoRaWAN-compatible network size estimation protocol that leverages the onboard Optimistic Collision Information (OCI) estimator.

Our solution, LoRa-OCI (L-OCI), was integrated into FLoRaSat, a C++ discrete-event DtS-IoT simulator that integrates realistic orbital and LoRa/LoRaWAN communication models. Through an extensive simulation campaign, we can determine appropriate LoRa configurations to achieve low root mean square error (RMSE) and low power consumption. Additionally, our results indicate that the approach is relatively insensitive to LoRa parameters when assessing the aggregated throughput of a Slotted ALOHA Game (SAG) protocol throttled by L-OCI.

TABLE OF CONTENT

Table Index

Illustration Index

Chapter 1

Introduction

Direct-to-Satellite Internet of Things (DtS-IoT) is a novel approach to integrated terrestrial and spatial wireless communications where small low-energy nodes on Earth directly communicate with a Low Earth Orbit (LEO) satellite [1]. The use of small satellites, including affordable CubeSats, enables the establishment of dependable and cost-effective networks by relaying packets to ground stations or other satellites in cases where a constellation is deployed [2]. This will lead to an expansion of the network's coverage and support. Conventional Low-Power Wide Area (LPWAN) technologies [3], such as LoRa, which are typically employed in urban and rural applications, can be modified to support DtS-IoT networks [4]. Furthermore, adopting open standards like LoRaWAN facilitates the unrestricted deployment of LoRa networks, as they comply with regional regulations. When coupled with CubeSats, this approach offers an economical alternative for building satellite networks that is competitive with available low-power, low-data rate satellite solutions (e.g., Argos [5], APRS [6], S-AIS [7], and ADS-B [8]).

One of the primary obstacles encountered in DtS-IoT networks involves the scalability of the network, particularly in situations where thousands of nodes must be served by a single satellite. Using Intelligent Medium Access Protocols (MAC) is critical in addressing the unavoidable packet collisions and enhancing the network's overall performance. Recent research has shown that incorporating information on the network size, i.e., the number of nodes under satellite coverage, into a purpose-built MAC protocol [9] can significantly enhance the network's performance in terms of throughput. However, obtaining this information is often challenging, as the nodes are frequently deployed in harsh, remote environments and may be isolated from one another [10].

This study aims to implement and evaluate a network size estimation mechanism for DtS-IoT using the LoRa/LoRaWAN protocol stack. We employ the Optimistic Collision Information (OCI) estimator introduced in [10] to achieve this goal. The selection of OCI is based on its superior performance in terms of low Root Mean Square Error (RMSE) estimation and power efficiency compared to other modern network size estimators. Nevertheless, the simulations conducted in [10] relied on simplistic assumptions without considering any underlying DtS-IoT protocol or communication model. The mechanism proposed in this thesis, named LoRa/LoRaWAN-based OCI (L-OCI), is designed to operate on a Frame Slotted ALOHA approach, employing LoRa/LoRaWAN framing and Chirp Spread Spectrum modulation with the corresponding Spreading Factors (SFs) described in the LoRa/LoRaWAN specification [11]. Consequently, the approach can be implemented on standard commercial IoT nodes. To assess the feasibility of OCI in a practical scenario, we offer a multi-objective parameter selection algorithm for L-OCI. Also, we implemented L-OCI and conducted a comprehensive simulation using the FLoRaSat simulator [12]. FLoRaSat was specifically developed to simulate DtS-IoT networks, incorporating LoRa-based networks and realistic orbital propagation and channel models.

1.1. Hypothesis

A network size estimation-based uplink transmission policy will improve the throughput of a dense DtS-IoT network while minimizing the energy consumption compared to the previous state-of-the-art models, closer to the omnipresent scheduler model who has complete traffic knowledge.

1.2. Objectives

- Develop and improve a DtS-IoT simulator (FLoRaSat) to generate reliable traffic pattern synthetic data
- Develop learning techniques to predict network size of end-devices
- Develop/Improve uplink transmission policies to rule channel allocation based on network size knowledge
- Train the proposed learning models on the synthetic data to create useful knowledge on the network in order to fit the best possible uplink transmission policies for the given configuration
- Test and analyze the behavior of the proposed policy based on the optimization objectives

1.3. Methodology

- *Protocol Models*: Engineer approach: detailed protocol models (i.e., physical, link and network layers) are implemented to study the expected performance and resource consumption of DtS-IoT by means of simulations in FLoRaSat. The model also includes a core network module for the architecture axis.
- *Abstract Models*: Computer Science approach: system-level models are developed to abstract the network elements and time-dependent resources to then enable optimized decision making based on Mixed Integer Linear Programming (MILP) or Markov Decision Processes (MDP) models.

1.4. Organization of the thesis

This thesis is organized as follows: Chapter 2 introduces the background topics and work laying the foundation of this thesis, in Chapter 3 the proposed technique L-OCI is described, Chapter 4 present the simulation tool FLRoRaSat used to evaluate the proposed L-OCI, then the results are presented and discussed in Chapter 5 and, finally, the conclusion of this thesis is given in Chapter 6.

Chapter 2

Background

In this Chapter the basic concepts of LoRa and LoRaWAN are presented. Then a brief introduction to space technologies is given, with emphasis on low cost spacecraft who are candidates to enable the extension of IoT to space. Finally, a detailed description of the OCI network size estimator is presented, laying a foundation for this thesis.

2.1. Long Range (LoRa)

Since LoRa is a proprietary radio communication technique, details on the exact modulation and demodulation process are not publicly available, yet it is one of the most popular wireless platforms for the IoT both in real life applications and research, together with NB-IoT.

The adaptability of LoRa to different scenarios makes it a promising candidate for the DtS-IoT paradigm. In fact, it has been achieved the world record distance of 832 kilometers for a LoRa packet transmission[1].

A typical LoRa radio comes with 5 configuration parameters

- Transmission Power (TP)
- Carrier Frequency (CF)
- Spreading Factor (SF)
- Bandwidth (BW)
- Coding Rate (CR)

The selection of configuration parameters will have a significant impact on the network. For instance, the transmission time of a payload of 10 bytes can be up to 50 times more depending on the specific set of parameters. The transmission time (or Time on Air) is of great importance when evaluating specific scenarios for LoRa-based IoT networks and it can be calculated according to the following set of equations:

[1] LoRa world record distance note.

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

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

$$n_{payload} = 8 + max\left(\left\lceil \frac{8PL - 4SF + 28 + 16CRC - 20H}{4(SF - 2DE)} \right\rceil (CR + 4),\ 0\right) \tag{2.3}$$

$$T_{payload} = n_{payload} \cdot T_{sym} \tag{2.4}$$

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

Equation 2.5 corresponds to the total Time on Air of a LoRa transmission, considering both the preamble and payload time (refer to the LoRAWAN specification [13] for a detail description of packets). Equations 2.1, 2.2, 2.3 and 2.4 depend on the following extra parameters (on top of the aforementioned configuration parameters):

- PL: payload size in byte
- H: header status, 0 is enabled and 1 is disabled
- DE: Low Data Rate Optimize, 0 is disabled and 1 is enabled
- CRC: Cyclic Redundancy Check, 0 is disabled and 1 is enabled
- $n_{preamble}$: number of symbols in preamble

It is also important to remember the Friis transmission equation for the specific case when considering a free space path loss 2.6. With this simplification on the proposed model it is possible to calculate the maximum range for a given combination of the following parameters:

$$\frac{P_r}{P_t} = G_t G_r \left(\frac{\lambda}{4\pi d}\right)^2 \tag{2.6}$$

- P_r: power at the receiver
- P_t: power at the transmitter
- G_r: receiver antenna gain
- G_t: transmitter antenna gain
- λ: wavelength of the carrier signal (related to CF according to $\lambda \cdot CF = c$)
- d: distance traveled by the signal
- c: speed of light on vacuum

LoRa operates on license-free frequency bands regulated by the corresponding spectrum agencies for each region of the world and, thus, restraining some of the configuration parameters aforementioned differently. In this work, the focus is on the regional parameters set for the European region.

2.2. Long Range Wide-Area Network (LoRaWAN)

LoRaWAN is the data link layer built on top of the physical layer provided by LoRa. The LoRaWAN specification [13] is developed and maintained by the LoRa Alliance, an open non-profit organization whose goal is to promote the use of LoRaWAN for IoT applications.

LoRaWAN defines three operation modes: Class A, Class B and Class C, in order to enhance adaptability of the network to different use scenarios. The communication strategy of each mode is depicted in figure 2.1.

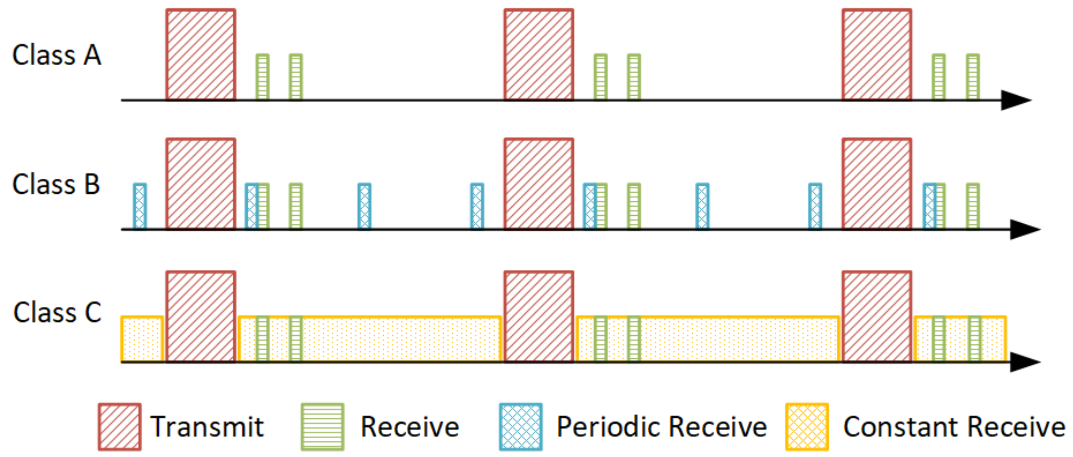


Figure 2.1: LoRaWAN Classes, image from [14]

In all three modes of operations, the node can begin an uplink transmission (in red) whenever it wants, respecting the restriction on the duty cycle given by the regional parameters. After a transmission, the node opens two reception windows (in green) in order to receive a downlink transmission sent by the network server through the designated gateway. This base operation mode constitutes the Class A mode, which is the most energy efficient one but restraining the node to receive downlink messages only after it has sent an uplink transmission, impacting negatively the downlink throughput specially when the usage scenario requires the node to constantly receive instructions from the network server.

In order to improve the downlink throughput, Class B operation mode defines periodic reception windows (in blue) allowing the gateway to transmit downlink messages even if the node does not have uplink messages to sent. This requires synchronization between the node and the gateway, achieved through a periodically scheduled beacon message broadcasted by the gateway. With this approach, the downlink throughput can be improved at the expense of energy efficiency due to the opening of extra reception windows.

Lastly, in scenarios where energy constraints are not a problem for nodes, Class C allows the node to receive downlink messages whenever it is not transmitting an uplink message (in yellow).

2.2.1. LoRaSync and Class S

The main issue with the classes defined in the LoRaWAN specification is the scalability of the network since a node can begin a transmission at any moment as the LoRaWAN specification implements a pure ALOHA protocol for the Medium Access Control [15]. A way to improve the uplink throughput is using a Time Division Multiple Access (TDMA) technique such as a slotted ALOHA, where a node can start a transmission only at the beginning of a time slot.

In [16], the authors propose the new *Class S* as an extension of LoRaWAN Class B, based on the slotted ALOHA technique and which they further improve in [17] by defining a beacon skipping mechanism and more robust slots when facing synchronization errors between node and gateway due to clock drift. This improved version, named *LoRaSync*, is described in [17].

Since *LoRaSync* is based on Class B, the periodic downlink reception windows are scheduled upon beacon reception. The beacon is also used to synchronize the uplink slots whose length is determined by the maximum clock drift δ_{max} and the maximum Time on Air ToA_{max}. The authors prove this MAC technique improves scalability of the network by reducing collision in the uplink channel and thus it has been chosen for the implementation of the OCI technique.

The slot time T_{slot} is defined by equation 2.7. The number of slots n_{slots} that can be fit inside the Beacon Window T_{window} is given by equation 2.9. The *floor()* function ensures the last slot will not overlap with the Beacon Guard.

$$T_{slot} = 2\delta_{max} + ToA_{max} \tag{2.7}$$

$$T_{beacon} = T_{reserved} + T_{window} + T_{guard} \tag{2.8}$$

$$n_{slots} = \left\lfloor \frac{T_{window}}{T_{slot}} \right\rfloor \tag{2.9}$$

2.3. Space technologies

2.3.1. Satellite Orbits

The DtS-IoT approach studied in this work is one where the IoT devices communicate with satellite located in the Low Earth Orbit (LEO), orbiting the Earth at around 600 kilometers of altitude and with speeds over the 7 kilometers per seconds. Compared to navigation satellites in the Medium Earth Orbit, such as GPS, or telecommunication satellites located at the Geostationary Orbit (see figure 2.2), LEO satellites have a much reduced view of the Earth and thus range. The distance to the horizon for a LEO satellite at 600 kilometers is of around 2800 kilometers. However, the proximity to Earth comes with reduced launching costs so deploying a constellation of LEO satellites can help solve the problem of coverage. In LEO we can find the International Space Station (ISS), the Hubble telescope and the Iridium satellite constellation (figure 2.2).

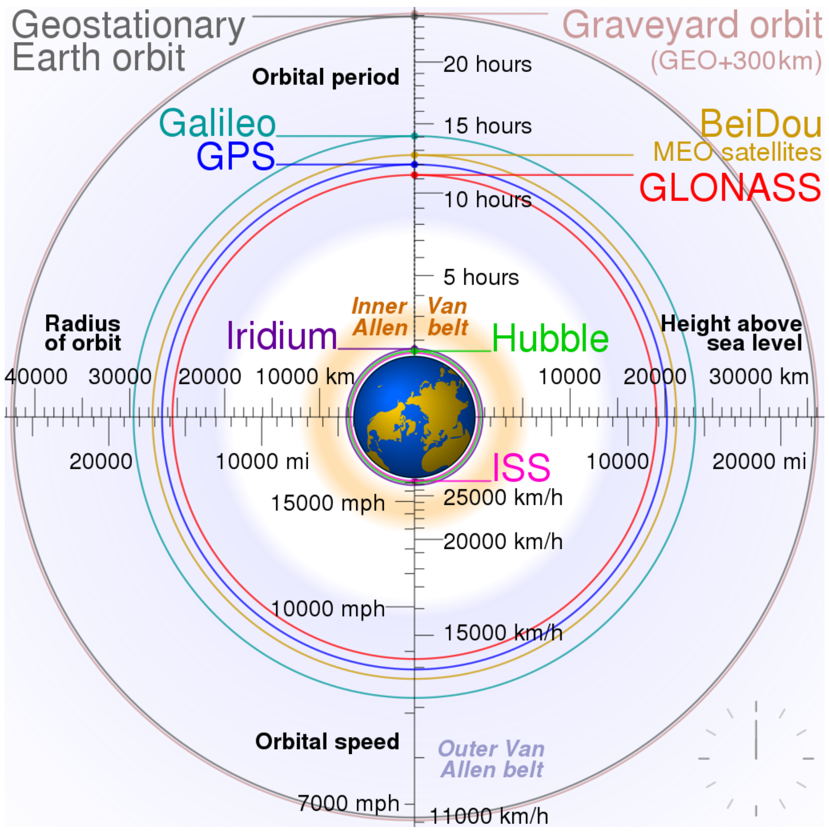


Figure 2.2: Satellite navigation orbits comparison, taken from Wikipedia

In order to predict a LEO satellite's orbit and accurately determine its position at a certain time, mathematical models have been developed and refined. In particular for LEO satellites, the orbital propagation model SGP4 [18] is widely used to this end. To obtain a more complete and accurate model of a DtS-IoT scenario, using SGP4 inside the simulation is desired.

2.3.2. The CubeSats Standard

The space sector has been revolutionized in the new millennium due to the development and rapid adoption of the so called CubeSat [19], a standardized nano satellite originally designed for LEO orbits. The CubeSat is composed by one or more basic units, a cube of a size 10x10x10 centimeters and a weight of around 1.33 kilograms. The reduced size and standard design helped to considerable decrease the costs and times of all the satellite development phases, from design, manufacture, and testing until launch in orbit. Its use as spatial vehicle for new technologies and space science has been adopted by all actors in both public and private sectors and prompted the inclusion of new actors such as universities and developing countries [20]. The CubeSat is then an ideal candidate for the deployment of DtS-IoT networks.

2.4. Optimistic Collision Information-Based estimator (OCI)

OCI [10] is a network size estimator for DtS-IoT networks designed under the principle of reducing the computational complexity required while maintaining a low estimation error compared to other State of the Art estimators. As mentioned in the previous Chapter 1, one of the main goals of this work is the implementation of the OCI estimator in the FLoRaSat simulator in order to test its performance on a more realistic scenario.

The OCI estimator was conceived to work based on a Frame Slotted ALOHA (FSA) logic with a beaconing system for time synchronization between the gateway and the nodes. The model is depicted in the figure 2.3, showing the slotted uplink. The assumptions in [10] are the following:

1. The communication between the nodes and the satellite follows a FSA logic
2. Nodes are randomly distributed under the satellite footprint. The set of nodes under the satellite footprint is called a cluster and there is one for each frame
3. A cluster is stable for the duration of a frame
4. The nodes do not remain idle, they will all transmit on each frame. By design, this is equivalent to assume all nodes in the cluster receive the corresponding beacon
5. Time synchronization between nodes and the satellite is available
6. Failures in decoding can either be due to collisions or to insufficient received signal power
7. Upon beacon decode, every node will choose a slot to transmit in with probability $\frac{1}{w}$, where w is the number of uplink slots of a frame

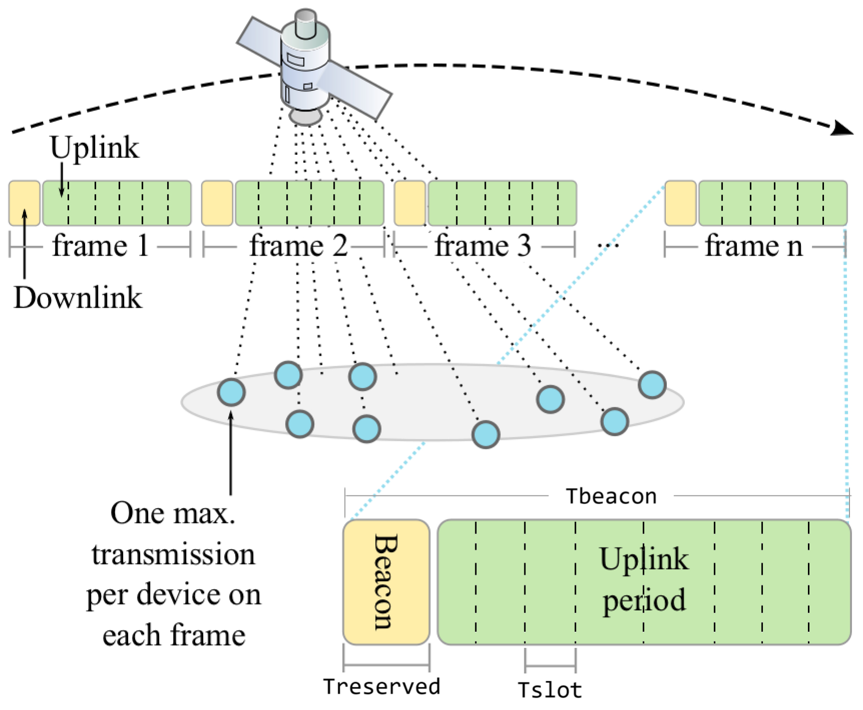


Figure 2.3: DtS-IoT network operating under a Frame Slotted ALOHA logic, adapted from [1]

Given a frame, the satellite determines the status of each uplink slot according to three possible values: 1. the slot is IDLE when there are no reception attempts, 2. the slot is SUCCESSFUL when there is a successful signal decode, and 3. the slot is COLLIDED when two or more nodes attempt to transmit on the same slot. It is important to notice that since transmission can also fail due to low power, the satellite can perceive a slot as IDLE even when two or more nodes chose to transmit on the same slot. The slot status data corresponds to the input data of the OCI estimator. For each frame, a tuple i, c, s is determined, where i is the number of IDLE slots, c is the number of COLLIDED slots, and s is the number of SUCCESSFUL slots.

The estimator operates in two phases. During the first one, named the estimation phase, slot status data is collected for a varying number of nodes. With this data, OCI performs a naive estimation of the actual number of nodes, N_n, under the footprint (or cluster size) of the satellite assuming no more than two nodes are involved in every collision. The naive estimation ϕ of N_n is calculated as $\phi = s + 2c$. Figure 2.4 shows the naive estimation as a function of the cluster size N_n for different frame lengths in terms of slots w. The naive estimation quickly underestimates the number of nodes when $N_n > w$ due to the assumption of neglecting high order collisions. Hence, the naive estimation always converges to $2w$.

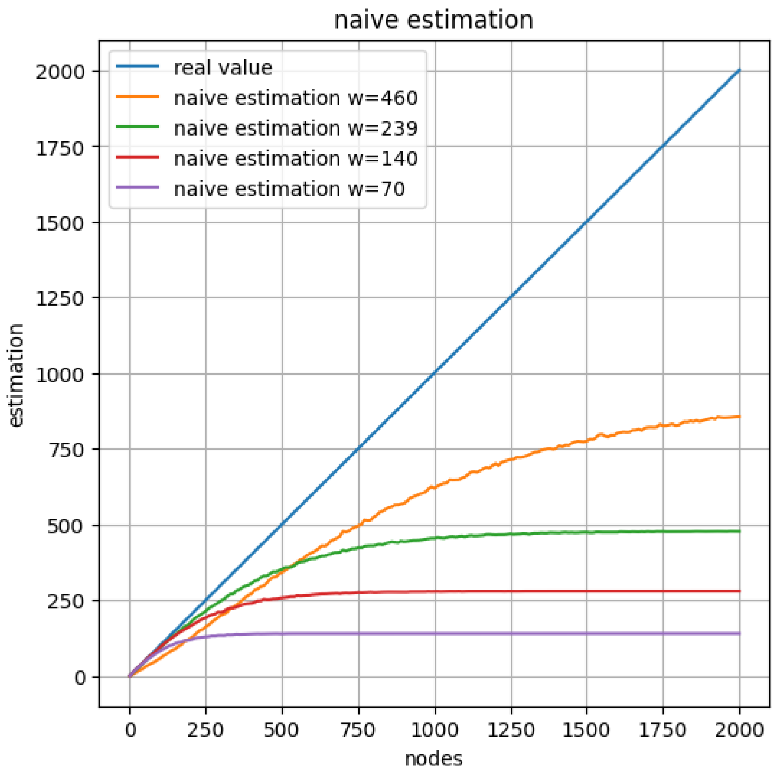


Figure 2.4: Naive estimations for frames with different number of slots w

To solve the underestimation without knowing the order of the collisions, the authors in [10] propose to create a function that maps the naive estimations curve into the real node value curve. This method requires bijectivity between the naive estimations and the real value of nodes and, therefore, the frame length w shall be long enough so that the naive estimation has no repeated values for the simulated domain. This function is created by means of two polynomial fits. The first one is used to smooth the naive estimation curve, while the second one maps the resulting curve of the first fit to the curve of the real number of nodes.

After determining the polynomial's coefficient of the second fit, the OCI mechanism enters to the operational phase, during which it applies the function to the naive estimation to get the OCI estimation of the real value. The algorithm for this phase is described in detail in [10].

Chapter 3

Lora-based OCI estimator (L-OCI)

One cannot directly use the OCI estimator as defined in [10] in LoRa-based networks without extending the MAC layer specified by LoRaWAN. Specifically, a protocol based on a TDMA technique with time-slotted frames and a time synchronization system must be added to the MAC layer. For L-OCI, we leverage LoRaSync [17], which we find is an appropriate Class B variant technique that satisfies the synchronization requirements of OCI and DtS-IoT. LoRaSync extends LoRaWAN Class B by adopting a beaconing mechanism that enables time synchronization and scheduling uplink slots during the beacon window. The structure of the beacon period is divided into three phases.

Phase 1: Beacon Reserved

During this phase, the gateway broadcasts the beacon, and ground nodes decode it. To optimize energy usage, it is important to avoid persistent reception in resource-constrained nodes. Therefore, it is necessary to use appropriate techniques, such as in [21], to estimate the satellite visibility. Then, a node may determine the optimal time to open the reception window. Once the beacon is received, the node will be synchronized on a frame level.

Phase 2: Beacon Window

Uplink slots (UL) are slotted in time during this phase. An UL includes a maximum clock offset threshold time δ_{max} at the beginning and at the end of the slot and a maximum time on air ToA_{max} in between, which depends on the LoRa parameters and the payload size. The parameter δ_{max} provides the required time synchronization contention guards for the beacon and clock drifts in DtS-IoT. Note that the downlink mechanism defined by LoRaWAN Class B still exists but is not depicted here since user data downlinks are unnecessary for L-OCI estimation. Every node who receives the beacon will transmit on a random slot, chosen following an uniform distribution.

Phase 3: Beacon Guard

Finally, the beacon period ends with the guard phase, allowing the gateway to decode the last transmissions.

At the end of the Beacon Guard, the L-OCI data is collected, which corresponds to the number of slots labeled as IDLE slots i, the ones labeled as SUCCESSFUL s, and the ones labeled as COLLIDED c, for a given number of nodes n attempting to transmit during the frame. A series of (n, i, s, c) tuples is obtained for increasing values of n to build the OCI training dataset, from which the naive estimation vector is obtained and then the OCI estimator is calculated. Having collected a dataset for training and another for the test phase for a maximum number of nodes N, the procedure to obtain the OCI estimation vector is detailed in algorithm 1.

Algorithm 1 details the OCI mechanism. The first step is to obtain the naive estimations from the training data set, then obtain the OCI estimator by means of two polynomial fits, and lastly, apply the estimator/polynomial to the naive estimation obtained from the test data set.

The number of slots w in the frame is a crucial parameter for L-OCI, as illustrated in Fig. 2.4. It is calculated as $w = \lfloor T_{win}/T_{slot} \rfloor$, where T_{win} is the beacon window time, and T_{slot} is slot duration. Therefore, w depends on the frame length, the maximum time-on-air (ToA_{max}), and the underlying LoRa parameters. The choice of SF and BW parameters of LoRa, and the frame length influence the non-trivial trade-off that determines the optimal operation of L-OCI. A higher SF (e.g., SF12) increases communication range and ToA_{max} but reduces the number of slots in the frame, which can limit the number of transmission opportunities. Conversely, a lower SF like (e.g., SF9) reduces both communication range and ToA_{max}, allowing more slots in the frame but reducing communication range. An optimal L-OCI parameter selection mechanism is proposed in section 3.2.

Algorithm 1: OCI algorithm

```
   Input: number of nodes $N$, training dataset $trainOCIdata$, test dataset
          $testOCIdata$
   Output: $OCIestimations$ vector
1  Initialize empty 1D arrays $trainNumNodes$ and $trainNaiveEstimations$;
2  for each $[n, i, s, c] \in trainOCIdata$ do
3      append $n$ to $trainNumNodes$;
4      append $(s + 2c)$ to $trainNaiveEstimations$;
5  end for
6  Adjust a 7-degree polynomial $p_1$ with $[trainNumNodes, trainNaiveEstimations]$;
7  $naiveSmooth \leftarrow p_1(trainNumNodes)$;
8  Adjust a 4-degree polynomial $p_2$ with $[trainNumNodes, naiveSmooth]$;
9  Initialize empty 1D arrays $testNumNodes$ and $testNaiveEstimations$;
10 for each $[n, i, s, c] \in testOCIdata$ do
11     append $n$ to $testNumNodes$;
12     append $(s + 2c)$ to $testNaiveEstimations$;
13 end for
14 $OCIestimation \leftarrow p_2(testNumNodes)$;
15 Return $OCIestimation$
```

3.1. Performance evaluation

A way to measure the performance of the estimation, the Root-Mean-Square Error (RMSE) is used, as proposed also by the authors of OCI. The RSME is calculated as

$$\epsilon_{rsme} = \sqrt{\sum_{k=1}^{K} \frac{\phi_k - n_k}{K}} \tag{3.1}$$

, where K is the number of clusters, n_k is the real number of nodes in the cluster, and ϕ_k is the estimation of n_k.

The performance of OCI is also measured in terms of the computational costs required to obtain the estimation ϕ_k during the estimation phase only, assuming the function calculated during estimation phase is already stored in the gateway. This aspect will not be considered in this study since the mathematical method underlying OCI remains the same.

The Slotted Aloha Game (SAG) [9] is a MAC protocol proposed for satellite networks using FSA communication, thus suitable for OCI and L-OCI. SAG gives a transmission probability to each node based on the number of slots and the currently active nodes in the network (estimated with OCI or L-OCI). If there are fewer nodes than slots, the probability is set to 1, meaning all nodes will attempt to transmit during the frame. If there are more nodes than slots, the probability is adjusted so that the number of nodes attempting to transmit is close to the number of slots. SAG aims to achieve the maximum theoretical throughput with arbitrarily large node populations. The throughput is calculated as the number of successful transmissions in a frame divided by the time length of the frame in slots.

3.2. Parameters selection

There are several parameters involved both in LoRa and LoRaSync. In this work, the parameters selected to be object of study are the Spreading Factor (SF), the Bandwith (BW), and the number of slots w within a frame. We propose a heuristic to determine the best combination of parameters $x = (SF, BW, w)$ for a given cluster size n_k. To quantify what would be an optimal solution we propose a Multi-objective Optimization Problem (MOP) based on two objectives for the OCI estimator; first, to minimize the energy expended to obtain the estimator (polynomial coefficients), and secondly, to minimize the estimation error of L-OCI. The problem is formalized as follows:

$$\text{minimize } \mathcal{F}(x) = (f_1(x), f_2(x))^T, \text{ s.t. } x \in \Omega \tag{3.2}$$

, where Ω is the decision space. In this MOP, the objective function $f_1(x)$ is defined as the total energy spend in decoding failed transmissions by the satellite, either because of a collision or insufficient received signal power, during the L-OCI estimation phase. On the other hand, $f_2(x)$ is defined as the estimation error achieved during the L-OCI test phase.

The calculation of $F(x)$ requires L-OCI to be implemented within the algorithm solving the MOP. For this purpose, L-OCI simulation is based on a geometric simplification of the DtS-IoT allowing the model to integrate space-time constraints for the transmissions, in contrast to the arbitrary effective detection parameter of the original OCI simulation used in [10].

Algorithm 2 described this model, where a number of nodes N at random positions compete for a frame composed by w slots. The output are the total lost transmissions, low power and collided and the successful, idle and collided slots. Algorithm 3 solves the MOP described in 3.2 for given a maximum expected network size and a deployment radius. The outputs are the average lost power and the estimation error.

Consider a single node deployed on ground and a satellite emitting its beacon at time $t = 0$. This scenario is depicted in figure 3.1, where the red circle represents the coverage of the satellite given a combination of LoRa parameters and the blue circle represents the node deployment area, both centered on the projection on Earth of the satellite.

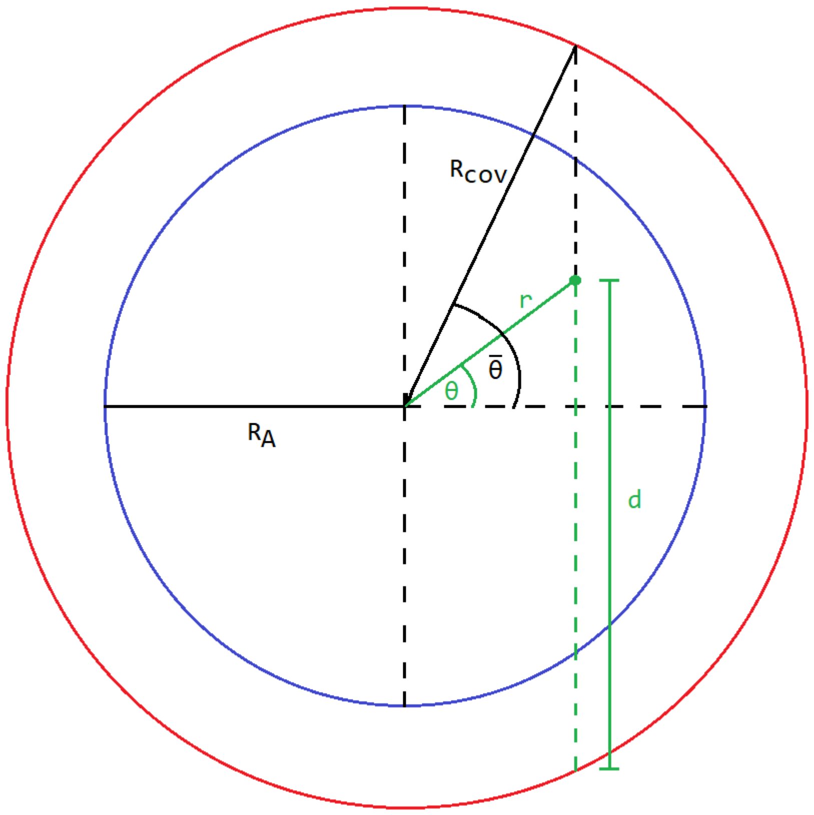


Figure 3.1: Simplified 2D scenario

The coverage radius $R_{cov}(SF, BW)$ does not depend on the number of slots w and it is calculated from the maximum range yielded by the combination of $\{SF, BW\}$. The node deployment radius R_A is such that $0 < R_A <= R_{cov}$. We consider a 2D space as a simplification of the 3D Earth model. Consider a node in position (r, θ) (in green) and assuming the satellite moves upwards at a constant speed v_s, the coverage time for the node can be estimated by equation 3.4

$$R_{cov} \cdot cos(\bar{\theta}) = r \cdot cos(\theta) \tag{3.3}$$

$$t_{support}(SF, BW) = \frac{d}{v_s} = \frac{R_{cov} \cdot sin(\bar{\theta}) + r \cdot sin(\theta)}{v_s} \tag{3.4}$$

On the other hand, following a FSA scenario where the node chooses a random slot $s \in [0, w]$, following an uniform distribution, to transmit upon beacon decode, the node would have finished its transmission at the beginning of the slot $s + 1$. The end of transmission time, t_{end}, can be calculated by equation 3.5 as follows.

$$t_{end}(SF, BW, w) = T_{reserved} + (s + 1) \cdot T_{slot}(SF, BW) \tag{3.5}$$

Finally, we define the case of failed transmission due to low received power if the condition given by equation 3.6 does not hold, so the node misses its oportunity to reach the satellite during coverage time.

$$t_{endTX}(SF, BW, w) < t_{support}(SF, BW) \tag{3.6}$$

The results of the MOP optimization are presented in Chapter 5. In the next Chapter 4, the FLoRaSat simulator is presented.

Algorithm 2: FSA simulation algorithm

```
Input: number of nodes $N$, node positions $positionData$, number of slots $w$, number
       of simulations $runs$, airtime $AT$, coverage radius $R_{cov}$
Output: average lost transmissions, average IDLE slots, average SUCCESSFUL;
        slots, average COLLIDED slots
1  $lowPowerRate \leftarrow 0$, $collidedRate \leftarrow 0$;
2  $avgIdleSlots \leftarrow 0$, $avgCollisionSlots \leftarrow 0$, $avgSuccessfulSlots \leftarrow 0$;
3  for $r = 0$ to $runs - 1$ do
4      $collided \leftarrow 0$, $lowPower \leftarrow 0$;
5      Initialize 1D array $transmissionsPerSlot$ with size=$w$ and all values set to 0;
6      for $(r, \theta) \in positionData$ do
7          Select a random slot $s \in [0, w-1]$ to transmit;
8          if $coverageTime(r, \theta, R_{cov}) <= endTransmissionTime(s, w, AT)$ then
9              $lowPower \leftarrow lowPower + 1$;
10         else
11             if $transmissionsPerSlot(x) > 0$ then
12                 $collided \leftarrow collided + 1$;
13             end if
14             $transmissionsPerSlot(x) = transmissionsPerSlot(x) + 1$;
15         end if
16     end for
17     for $j = 0$ to $w - 1$ do
18         if $transmissionsPerSlot(j) == 0$ then
19             $avgIdleSlots \leftarrow avgIdleSlots + 1$;
20         else if $transmissionsPerSlot(j) == 1$ then
21             $avgSuccessfulSlots \leftarrow avgSuccessfulSlots + 1$;
22         else
23             $avgCollisionSlots \leftarrow avgCollisionSlots + 1$;
24         end if
25     end for
26     $lowPowerRate \leftarrow lowPowerRate + (lowPower/N)$;
27     $collidedRate \leftarrow collidedRate + (collided/N)$;
28 end for
29 $avgLost \leftarrow lowPowerRate + collidedRate$;
30 Return $[avgLost, avgIdleSlots, avgSuccessfulSlots, avgCollisionSlots]/runs$
```

Algorithm 3: $F(x)$ algorithm

```
    Input: number of nodes $N$, deployment radius $R_A$, number of simulations $runs$,
           $nodeStep$, decision vector $x$
    Output: $F(x) = (f_1(x), f_2(x))$
 1  Initialize N node positions $(r_i, \theta_i)$ randomly and uniformly over the area defined by
      $R_A$ and store data in a 2D array $trainNodePositions$;
 2  Initialize empty arrays $trainOCIdata$ and $testOCIdata$;
 3  Repeat the previous step to get the test set $testNodePosisions$;
 4  $SF \leftarrow x(0)$, $BW \leftarrow x(1)$, $w \leftarrow x(2)$;
 5  $AT \leftarrow getAirtime(SF, BW)$;
 6  $R_{cov} \leftarrow getCoverageRadius(SF, BW)$;
 7  $nodes \leftarrow nodeStep$;
 8  while $nodes <= N$ do
 9      simulate FSA with $nodes$, $trainNodePositions$ and append to $trainOCIdata$;
10      simulate FSA with $nodes$, $testNodePositions$ and append to $testOCIdata$;
11      $nodes \leftarrow nodes + nodeStep$;
12  end while
13  $avgLost \leftarrow$ lost transmissions value for $N$ nodes from $trainOCIdata$;
14  $avgLostPower \leftarrow transmissionPower \times AT \times avgLost$;
15  $rsme \leftarrow getRSME(trainOCIdata, testOCIdata)$;
16  Return $(avgLostPower, rsme)$
```

Chapter 4

The FLoRaSat simulator

4.1. Overview

Simulations are key to test any theoretical models, the more precise the simulation is, the better is the validation of the assumptions made. This is in line with the objective of this work which is, as mentioned before, test the OCI estimator in a more realistic scenario.

4.1.1. The OMNeT++ and INET frameworks

OMNeT++ [22] was the chosen framework to simulate DtS-IoT networks due to its extensive use in research in the field of networks and active community [23]. As described in [23], it "is an extensible, modular, component-based C++ simulation library and framework, primarily for building network simulators". Moreover, on top of OMNeT++ there is the INET framework [24], "an open-source OMNeT++ model suite for wired, wireless and mobile networks", which has grown large enough over the years to be one of the essentials libraries for building network simulators in OMNeT++. It contains models for the Internet stack, routing protocols, link layer protocols, support for the wireless physical layer, node mobility, visualization and more. Both frameworks provide extensive manuals and guides which facilitate the development and provide insightful tutorials.

4.1.2. FLoRa, OS3, and leosatellites frameworks

FLoRa is an open-source framework for LoRa simulations in OMNeT++ [25]. It implements the physical and medium access control layers of LoRa and so, it provides the bedrock for the FLoRaSat simulator. The architecture of FLoRaSat is largely based on FLoRa but various modifications had been done in order to adapt it to simulate a DtS-IoT scenario since FLoRa was designed and tested on traditional IoT scenarios, i.e. terrestrial deployments where the LoRa gateway is on a fixed position and has no energy or communication constraints.

Because of the previous remark, a mobility model for the gateway had to be implemented in OMNeT++ to simulate the orbital dynamics of a satellite. To solve this issue, FLoRaSat relies on two other frameworks which implement the orbital dynamics models in OMNeT++. The first one is the Open Source Satellite Simulator (OS3) [26], where the authors implement the SGP4 and SDP4 orbital propagation models and provide the necessary modules to define orbits of objects given their TLE data. There were two main issues encountered with this framework during its integration to build FLoRaSat; first, OS3 was implemented on the outdated INET2 version, and second, it did not provide a mean to define orbits through their orbital parameters.

The issues with the OS3 framework were addressed in [27]. The authors provided an updated and extended version up to date with the current INET4. The leosatellites framework provides a mean to define orbits through its orbital parameters while still supporting the TLE orbit modules. Note that the scenario studied by the author in [27] focuses on Inter Satellite Links (ISL), i.e. communication between satellites, and on massive satellite constellations defined by the number of orbital planes and the number of satellites on each plane.

Summarizing, FLoRaSat is built on top of this three frameworks, and all of them rely on the INET framework. But still, a lot of effort in adapting each framework to support a DtS-IoT scenario and also in unifying all of the modules to work together was needed. The result is an open-source powerful simulation tool providing a wide range of research and test opportunities on different topics concerning DtS-IoT scenarios. Some of the supported functionalities are the following:

- Physical layer: opportunity to develop and test new modulation techniques such as the LR-FHSS, designed to address long range communication like satellite links [28]
- Link layer: novel medium access protocols can be implemented and studied, for both nodes and gateways
- Network and Transport layers: routing algorithms for satellite constellations can be implemented and tested
- Application layer: support of end applications on the node and sever part of a LoRaWAN network
- Channel models: testing of more accurate channel models is an important aspect to consider given the long distance links through the atmosphere in DtS-IoT

4.2. FLoRaSat architecture

This section presents a more detailed but not extensive description of the FLoRaSat framework focused on the main interacting modules on a typical Dts-IoT scenario plus a short simulation guide.

4.2.1. Physical layer implementation

The physical layer in FLoRaSat corresponds to the LoRa physical layer implemented in FLoRa. The original source code required little to no modifications. The layer is depicted in the figure 4.1. In this diagram, there is a distinction between .ned modules (rectangles with round corners) and C++ classes (ovals).

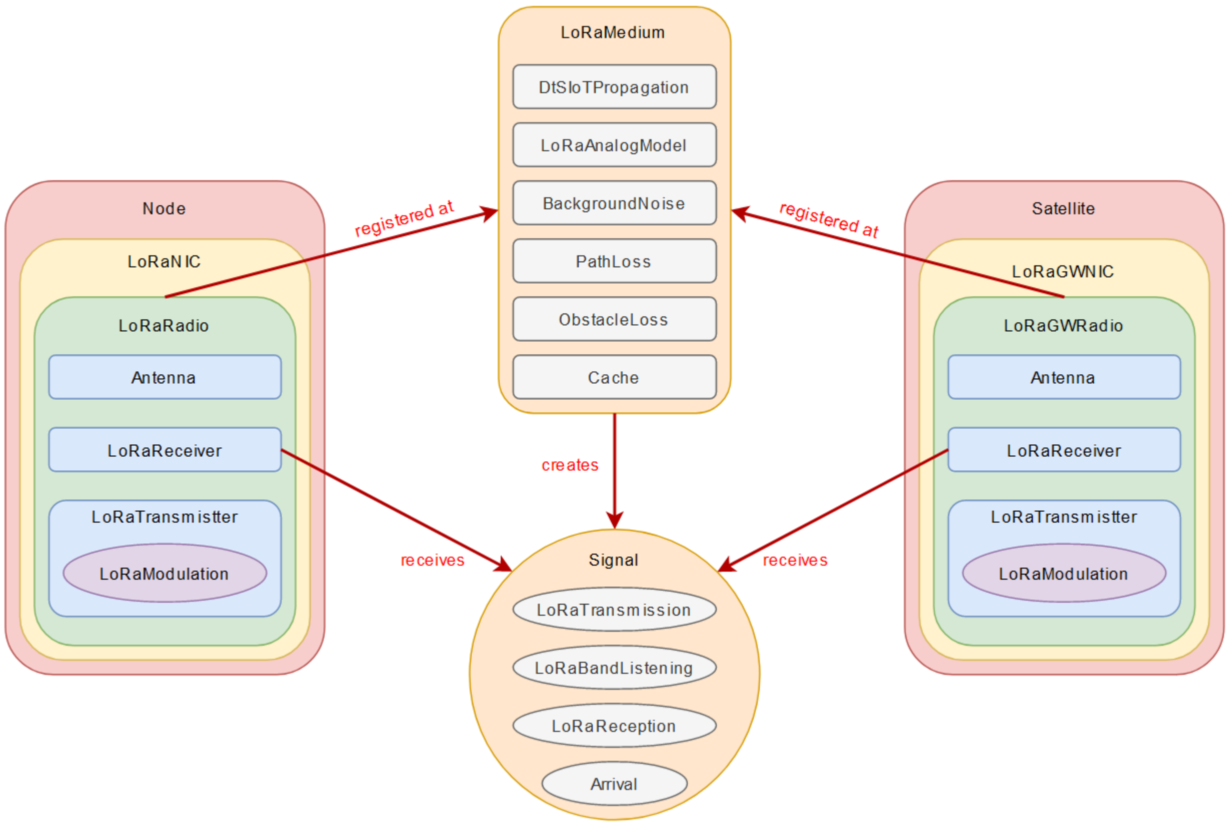


Figure 4.1: LoRa physical layer diagram in FLoRaSat

The main interacting modules in this layer are the Node and Gateway radios (LoRaRadio and LoRaGWRadio, respectively) and the medium module (LoRaMedium). Both radio modules contain three submodules, the antenna module, the receiver module (LoRaReceiver) and the transmitter module (LoRaTransmitter), and inside the latter is where the modulation takes place, defined by LoRaModulation. The channel module or medium module (LoRaMedium) contains various submodules defining the channel properties and other cache modules.

An end-to-end LoRa communication follows these steps:

1. Before the simulation starts, radios are registered at the LoRaMedium module who keeps track of their positions.

2. The LoRaTransmitter creates a transmission (LoRaTransmission) and sends it to the broadcast channel.

3. The LoRaMedium computes when and where the transmission will reach the potential receivers and how they will receive it. The module applies the channel properties to the signal and creates a Signal object for each receiver. The Signal object encompasses all the necessary information to process a reception.

4. The LoRaReceiver module receives the Signal object from the medium and process the information.

The LoRaMedium module is of great importance since it not only simulates the channel properties but also manages all the transmissions taking place during the simulation. During the implementation of FLoRaSat, this module required a major adaptation to support the new mobility module defining the movement of gateways (satellites).

4.2.2. Link layer implementation

The physical layer diagram in figure 4.1 shows the radio modules LoRaRadio and LoRaGWRadio are in fact submodules of the Network Interface Controller LoRaNIC and LoRaGWNIC modules, respectively. The second submodule of each NIC module is the Medium Access Control (MAC) module. The MAC modules implement the link layer defined by LoRaWAN for both the nodes (LoRaMac) and the gateways (LoRaGWMac).

The FLoRa framework provides the implementation of LoRaWAN classes A and B on the node side. However, the implementation of the Class B, on the gateway side, is not complete since it lacks support for the scheduled downlink slots. Nevertheless, the module does provide support for beaconing mechanism. This issue has not been addressed yet in the FloRaSat implementation but it does not constitutes a problem for this work since the OCI estimation is based only on uplink communication. A brief description on the implementation of Classes A and B provided by FLoRa and the proposed Class XS is given next, focused on the node side.

4.2.2.1. LoRaWAN Class A

The internal definition of each Class is by means of a Finite State Machine (FSM), implemented using the FSM library provided by INET. Figure 4.2 shows a flow chart diagram describing the Class A on the node side.

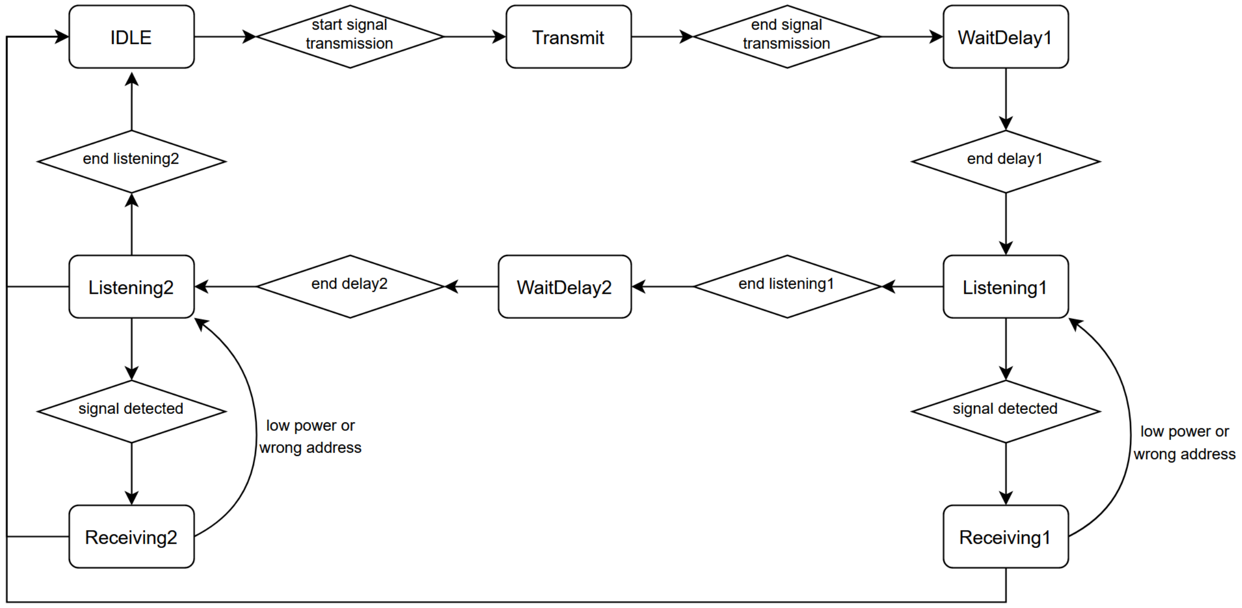


Figure 4.2: LoRaWAN Class A flow chart

On the gateway side, an FSM is not needed since the logic is simpler. The gateway duty is to always listen for incoming uplink transmission and to forward the traffic to the network server.

4.2.2.2. LoRaWAN Class B

Figure 4.3 shows a flow chart diagram describing the Class B on the node side. It is an extension of the FSM describing Class A as specified by the LoRaWAN standard [14].

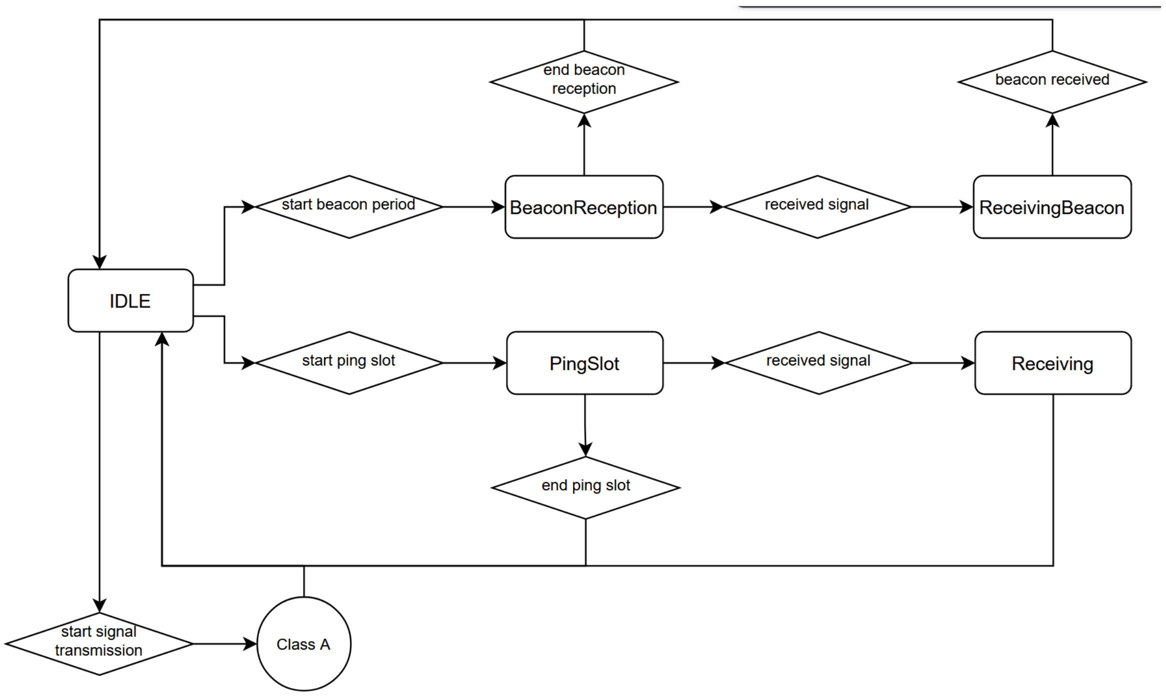


Figure 4.3: LoRaWAN class B flow chart

Again, an FSM is not needed on the gateway side. The gateway's duties are two: first, to broadcast periodic beacons, and second, set the radio to reception mode whenever it's not transmitting, and then forward the uplink traffic to the network server.

4.2.2.3. LoRaWAN Class XS

In order to perform the OCI estimation mechanism in FLoRaSat, this works includes the implementation of a modified version of the Class S proposed in [16]. The new version, named Class XS, implemented as an extension of the Class B provided by FLoRa, which lacks the downlink support. Figure 4.4 shows a flow chart diagram describing the Class XS on the node side.

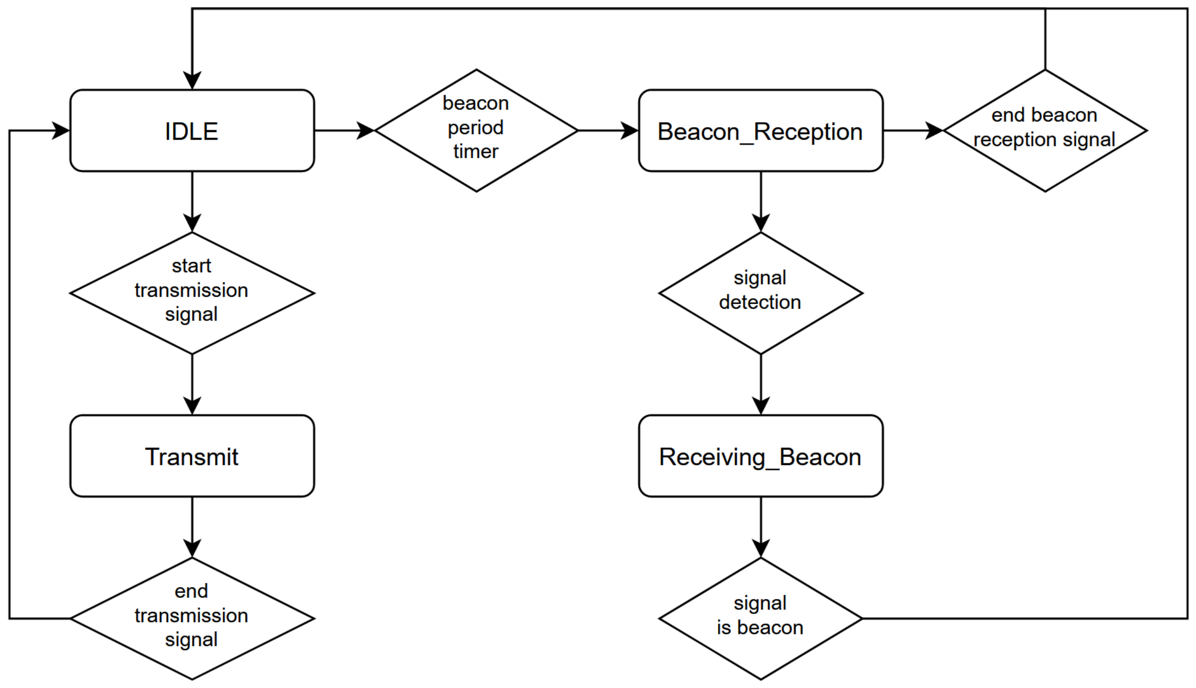


Figure 4.4: LoRaWAN Class XS flow chart

Once again, FSM was not required on the gateway side. However, at the end of each uplink slot, the gateway collects the core data for the L-OCI mechanisms, recording the number of successfully decoded signals, the total number of reception attempts, and the number of received signals with power below the gateway's sensitivity.

4.2.3. Satellite orbit definition

A single satellite or a constellation can be deployed in FLoRaSat by defining the orbital parameters describing the orbit. The parameters are given in the initialization file and during simulation the orbital propagator model is executed. The definition is straightforward and the parameters can be directly obtained from a TLE [29] or by orbit design.

4.2.4. Network architecture

A network is the main module of an OMNeT++ simulation. It defines the modules present in the environment, the connections between them by means of a channel module or the medium module where they can propagate their wireless transmissions. A generic DtS-IoT network is shown in Figure 4.5. Several networks can be included in a single initialization file.

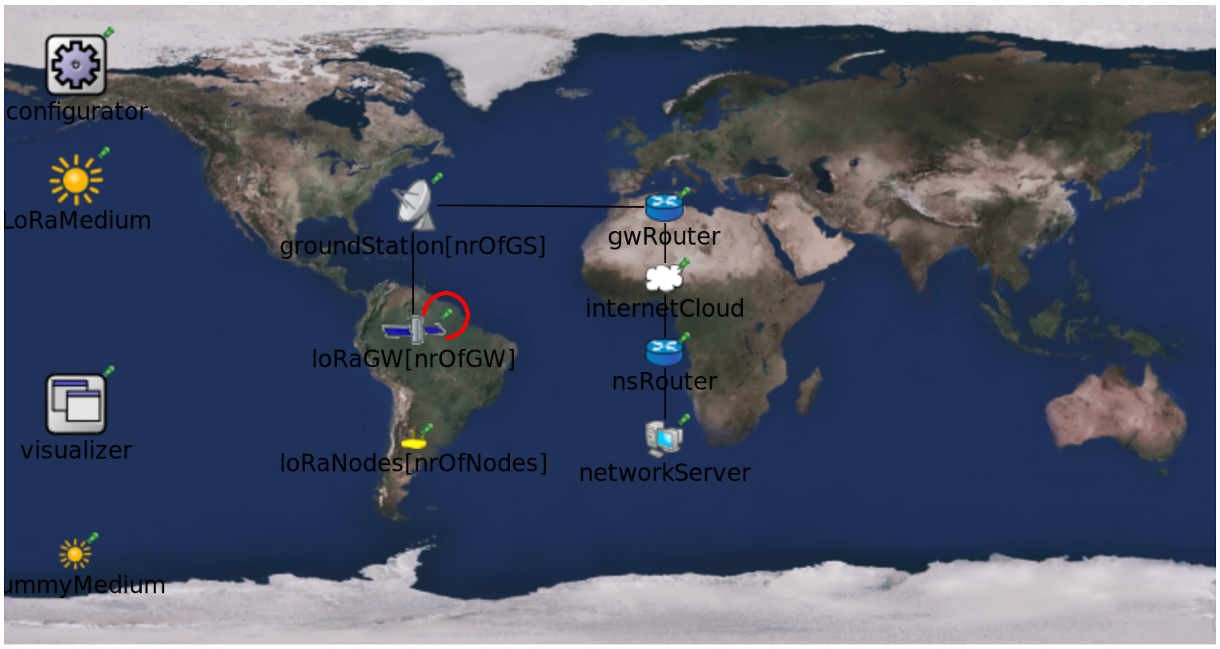


Figure 4.5: DtS-IoT scenario scheme in FLoRaSat

Chapter 5

Simulations and results

This section presentes the general simulation scenario used as a base for all the experiments performed in this thesis. It also presents the suite of experiments whose main objective is to asses the impact of the LoRa parameters selection on the network size estimation and the network performance.

5.1. Simulation scenario

The basic scenario is depicted in figure 5.1. We focus on a single satellite acting as a LoRa gateway relaying all packets sent from the nodes on the ground to the network server through the ground station. However, the satellite to ground station link is modeled as simple link with a fixed delay of 20 milliseconds and no channel saturation since it is not the focus of this work.

The satellite's orbit is a typical polar LEO orbit with an inclination of 98 degrees and an altitude of around 600 kilometers. The nodes are deployed randomly and uniformly over a circular region given a center in latitude, longitude coordinates, and a deployment radius. As mentioned in the previous sections, the L-OCI estimation requires a slotted communication and thus, the nodes and gateway use the Class XS operation mode for LoRa communication described in Chapters 2 and 4. Taking this into consideration, the scenario is set such that the satellite is positioned just above the deployment center at the beginning of the simulation. For all scenarios, the deployment radius is 475 kilometers, which is enough to cover the nodes deployed on ground when using the configuration parameters yielding the smallest coverage radius used in the simulations present in this chapter (i.e SF9 BW125).

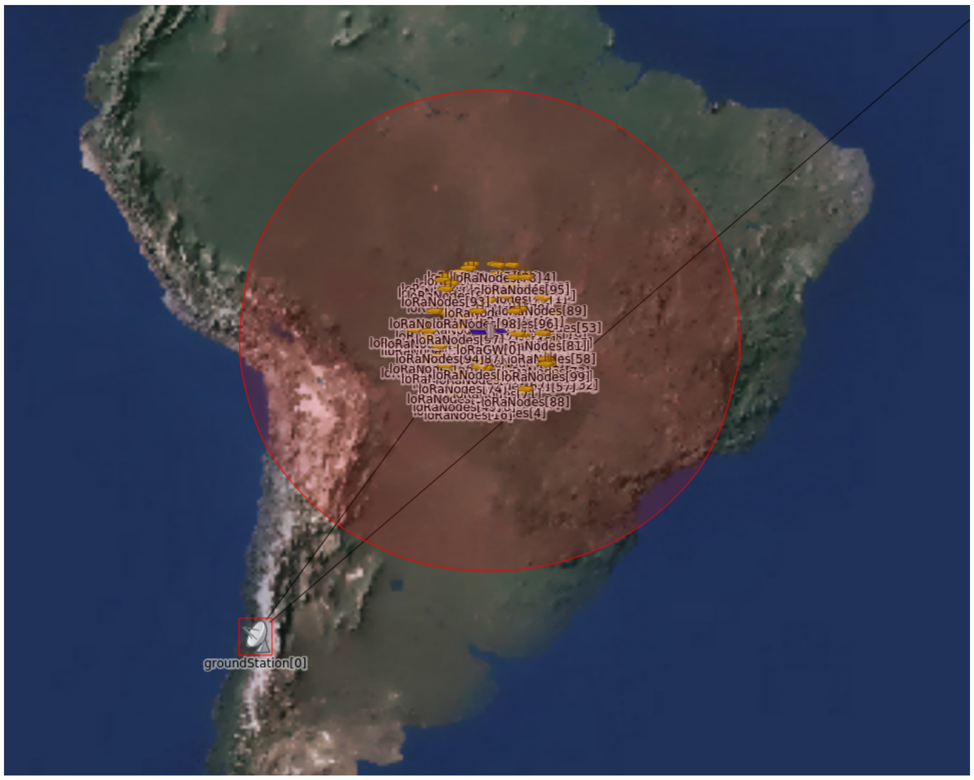


Figure 5.1: DtS-IoT study scenario

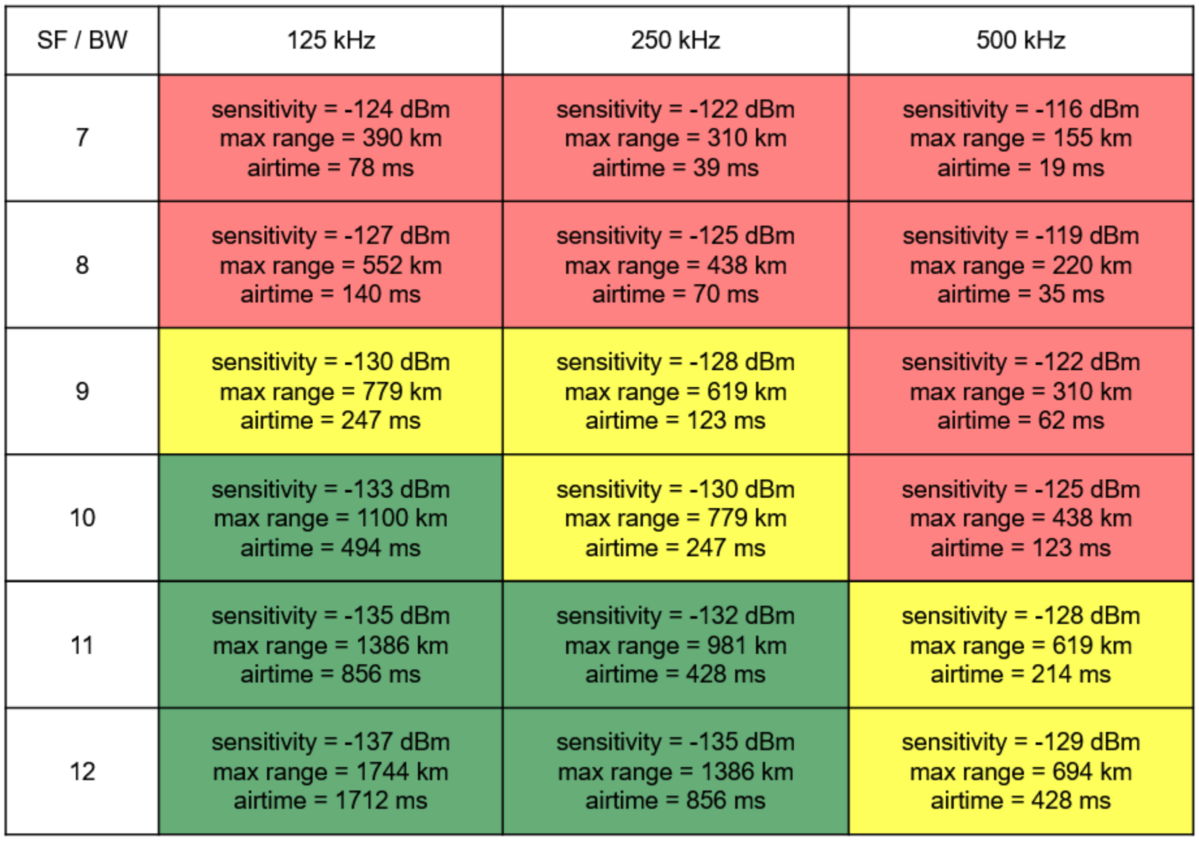
As for the LoRa parameters, the focus of this work is on variation of Spreading Factors (SF) and Bandwidths (BW), while the other parameters were selected following the specifications given by LoRaWAN for the European region [30] [2]. The fixed parameters are shown in the table 5.1.

Tabla 5.1: DtS-IoT study scenario global parameters [30]

Parameter	Value
Node antenna gain	5 [dBi]
Satellite antenna gain	0 [dBi]
Node Transmission Power (TP)	14 [dBm]
Satellite Transmission Power (TP)	3.5 [dBm]
Center Frequency (CF)	863 [MHz]
Coding Rate (CR)	4
Payload size (PL)	20 [bytes]
Beacon Reserved Time ($T_{reserved}$)	2.12 [s]
Beacon Guard Time (T_{guard})	3 [s]
Max Clock Drift (δ_{max})	0.01 [s]

[2] even though in the depicted scenario the simulation is performed over the South America region, this does not interfere with the European regional parameters for LoRa

Based on these parameters and using the equations provided in 2.1 (i.e. assuming a free space path loss channel model) it is possible to calculate the time on air (*airtime*) of a LoRa transmission and the maximum range of communication (*range*) for a certain combination of (SF,BW) parameters. The sensitivity values were obtained from a typical LoRa transceiver [31]. The *airtime* and *range* are shown in figure 5.2.



SF / BW	125 kHz	250 kHz	500 kHz
7	🟥 sensitivity = -124 dBm max range = 390 km airtime = 78 ms	🟥 sensitivity = -122 dBm max range = 310 km airtime = 39 ms	🟥 sensitivity = -116 dBm max range = 155 km airtime = 19 ms
8	🟥 sensitivity = -127 dBm max range = 552 km airtime = 140 ms	🟥 sensitivity = -125 dBm max range = 438 km airtime = 70 ms	🟥 sensitivity = -119 dBm max range = 220 km airtime = 35 ms
9	🟨 sensitivity = -130 dBm max range = 779 km airtime = 247 ms	🟨 sensitivity = -128 dBm max range = 619 km airtime = 123 ms	🟥 sensitivity = -122 dBm max range = 310 km airtime = 62 ms
10	🟩 sensitivity = -133 dBm max range = 1100 km airtime = 494 ms	🟨 sensitivity = -130 dBm max range = 779 km airtime = 247 ms	🟥 sensitivity = -125 dBm max range = 438 km airtime = 123 ms
11	🟩 sensitivity = -135 dBm max range = 1386 km airtime = 856 ms	🟩 sensitivity = -132 dBm max range = 981 km airtime = 428 ms	🟨 sensitivity = -128 dBm max range = 619 km airtime = 214 ms
12	🟩 sensitivity = -137 dBm max range = 1744 km airtime = 1712 ms	🟩 sensitivity = -135 dBm max range = 1386 km airtime = 856 ms	🟨 sensitivity = -129 dBm max range = 694 km airtime = 428 ms

Figure 5.2: Comparison of SF/BW configurations

Considering a satellite deployed at 600 km, the (SF,BW) configurations shown in red are those with a range shorter than the altitude of the satellite and thus they represent an impossible configuration. The ones in yellow are possible configurations but with limited range, while the ones in green are preferable configurations in terms of communication range. Therefore, in the following experiments, the red configurations are discarded as well as the column corresponding to a bandwidth of 500 kHz because it is not supported in Europe by the LoRaWAN specification [30]. Additionally, all nodes and the gateway use the same configuration parameters on a single simulation run.

In addition to the (SF,BW) parameters for the LoRa modulation, the third parameter under study is the beacon period time (T_{beacon}, here as BCN). When fixing the values for (SF,BW) then the number of slots w in the frame is given by equation 2.9. For a given number of nodes competing in a single frame, the more slots are available, the less collisions will occur. However, given the high speed satellite's orbit, the longer the frame is, the higher the probability of packets loss due to low sensitivity will be.

In summary, the control variables of the analysis are the spreading factor (SF), the bandwidth (BW) of LoRa in kHz, and the beacon period (BCN) in seconds, in addition to the network size. At the same time, the remaining parameters comply with the LoRa Alliance specification for the European region [30]. We determine the time on air (*airtime*) and the maximum range of communication (*range*) for a LoRa transmission based on the values of (SF, BW). The channel model assumed is a free space path loss model, and we consider the received sensitivity values from a typical LoRa transceiver, Semtech's SX1272 [31]. We run 30 simulation repetitions with different seeds for each input set (SF, BW, BCN).

As mentioned in section 2.4, the input data for L-OCI is the slot status count calculated for a frame. Figure 5.3 shows an example of the status of each slot for a single frame and Table 5.2 shows the logic for determining the slot status.

Tabla 5.2: Slot status

.	successful == 0	successful == 1
attempted == 0	IDLE (case 0)	impossible
detectable == 0	IDLE (case 1)	impossible
detectable == 1	COLLISION (case 4)	SUCCESSFUL (case 2)
detectable >= 2	COLLISION (case 4)	COLLISION (case 3)

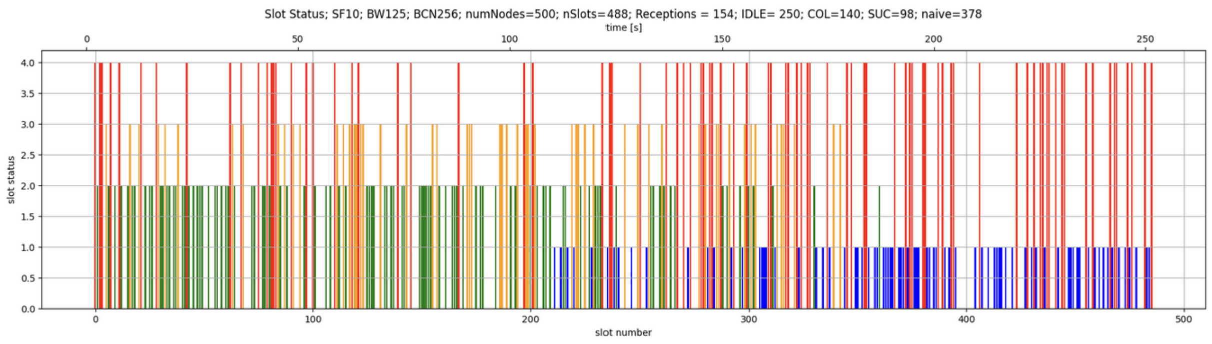


Figure 5.3: Slot status calculated for a frame

According to table 5.2, there are two possible cases for a single slot: in the fist case, no transmission was decoded successfully (successful==0), whereas in the second case one and only one transmission was decoded (successful==1). The different causes for this two possibilities are explained as follows. If there are no reception attempts during the slot time (attempted==0), then the slot is IDLE. The rest of the cases assume that there is at least one reception attempt. If there are no detectable transmissions (detectable==0), meaning that all arriving transmission have insufficient power to be decoded, then the slot is IDLE. Note that for both of the previous cases it is impossible to have a successful decoding.

If there is only one detectable transmission (detectable==1) then the slot is in COLLISION (case 4) if no transmission was decoded during the slot but if there was, then the slot is SUCCESSFUL. Lastly, if there are more than two detectable transmission then the slot is in COLLISION (case 4) if no transmission was decoded during the slot but if there was, then the slot is also in COLLISION (case 5). We make the distinction between the two cases with COLLISION status to emphasize the capability of the FLoRaSat simulator of modeling the capture effect by decoding the first signal to arrive to the gateway, since all nodes share the same LoRa parameters, then it is also the most powerful signal, while the following transmissions to arrive are discarded.

5.2. Simulation Results

5.2.1. Testing L-OCI limits

In a first experiment, we use the parameters $(SF = 9, BW = 125, BCN = 256)$, yielding $w = 939$ slots in the frame, and we vary the number of nodes from 0 to 2000 in increments of 40 nodes. The transmissions outcome is shown in Figure 5.4, with each possible outcome as the ratio to the total amount of transmissions (or node number since they all transmit once). It is evident how the long beacon period impacts the network given the large amount of lost transmissions due to low power.

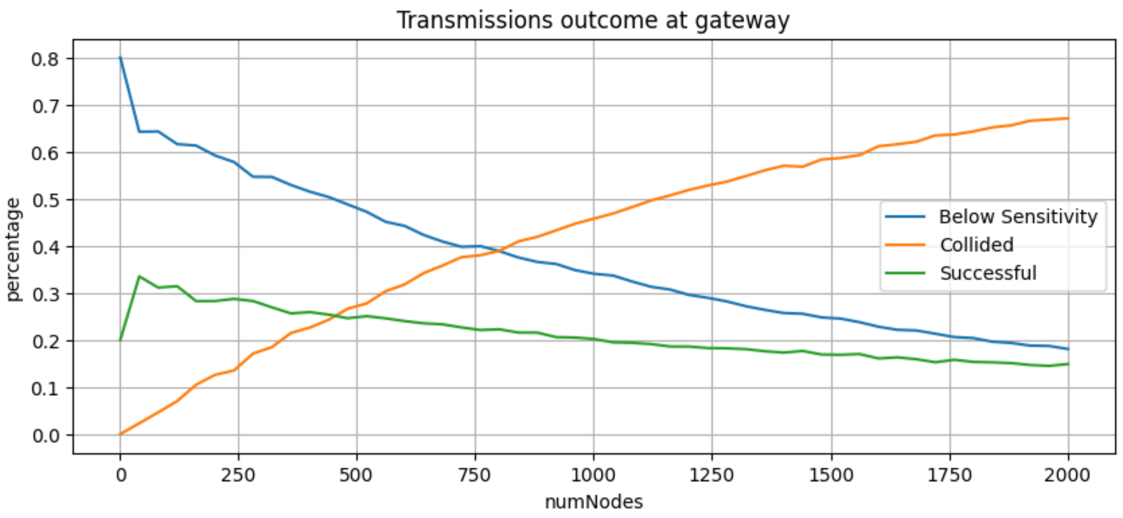


Figure 5.4: Transmissions outcome

The naive estimation and naive smoothed curves are shown in figure 5.5. Is is noted how the below-sensitivity transmissions impact the naive estimation since the IDLE slot are not considered, producing a significant underestimation for smaller networks.

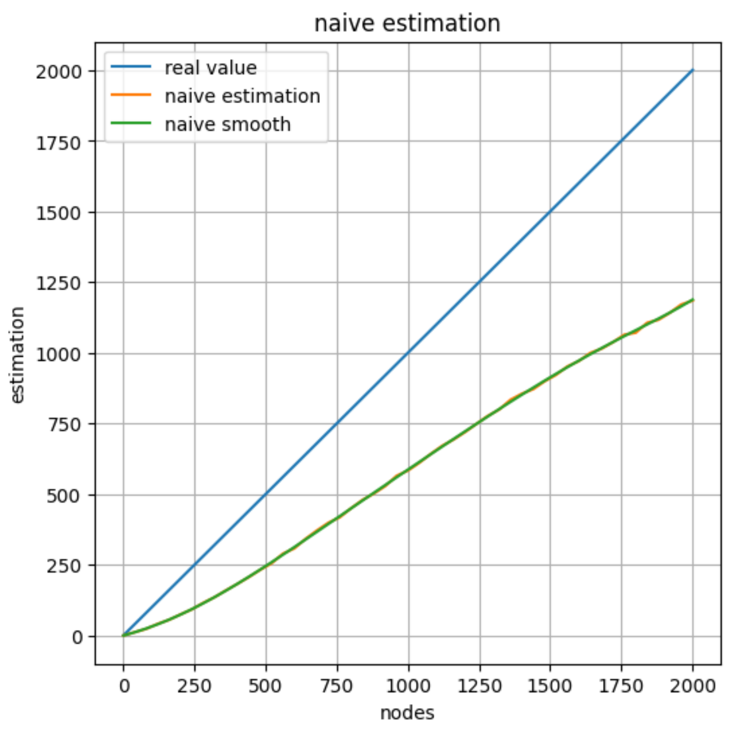


Figure 5.5: Naive estimation

For the test phase, we vary the number of nodes from 5 to 2045 in increments of 60 nodes, while the input set (SF, BW, BCN) remains the same. The L-OCI estimations obtained are shown in figure 5.6 and compared to the naive curve for the test phase. The estimation error in terms of RSME for the OCI curve compared to the real value curve is equal to 8.56.

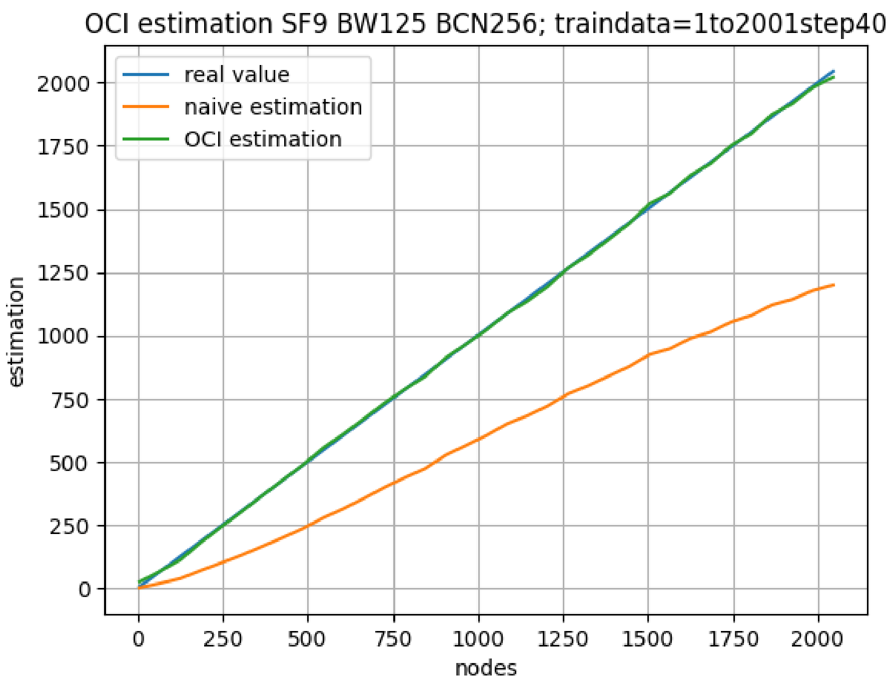


Figure 5.6: L-OCI estimations

It is noted how L-OCI achieves a low estimation error even in scenarios with low detection rate. This result was also stated by the original authors of OCI but their results only considered up to 25% packet loss [10].

Now, we consider a scenario that does not meet the bijectivity requirement for L-OCI. We consider the parameters ($SF = 9, BW = 125, BCN = 256$) but for the estimation phase we vary the number of nodes from 1 to 1001 in increments of 40 nodes and then, for the test phase, we vary the number of nodes from 5 to 2045 in increments of 60 nodes. The L-OCI estimations obtained are shown in figure 5.7. As expected, L-OCI fails to estimate outside of its domain, which confirms the need of a bijective function.

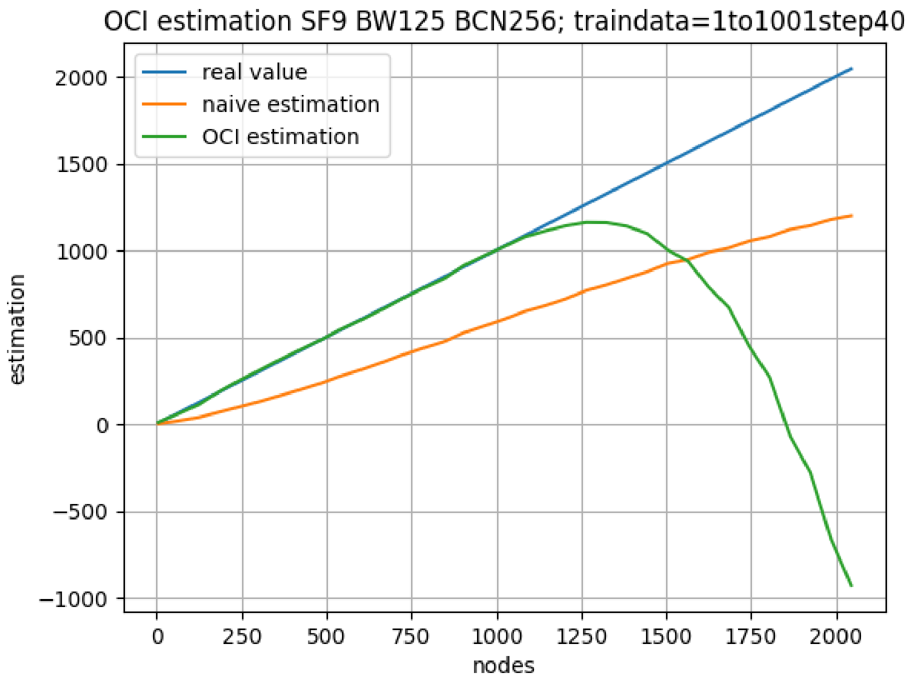
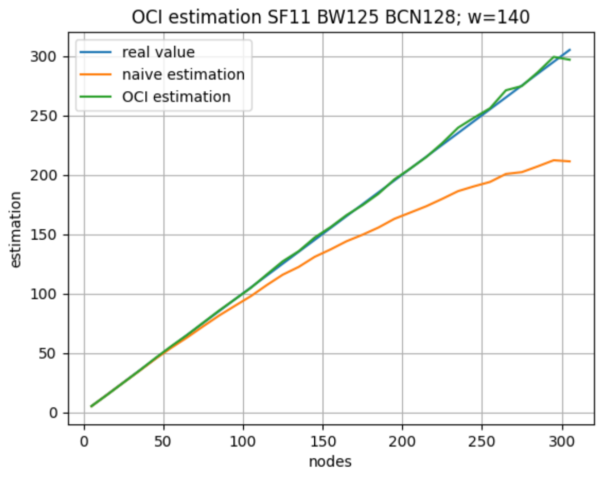


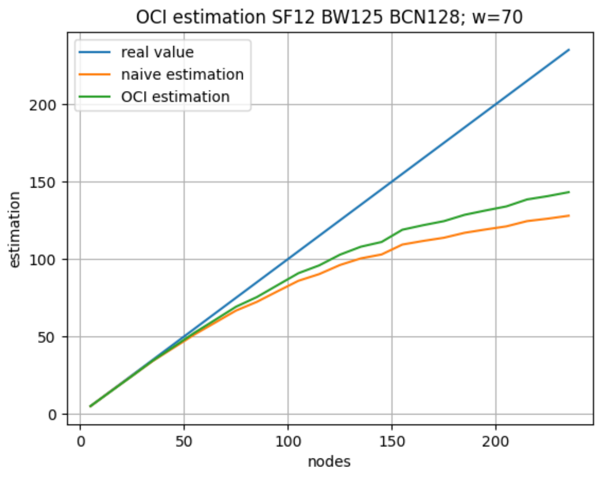
Figure 5.7: L-OCI estimations for experiment a

In a second experiment we show the impact of using an L-OCI estimator trained on a different number of slots w. For the estimation phase we consider the parameters ($SF = 11, BW = 125, BCN = 128$) , yielding $w = 140$ slots in the frame, and we vary the number of nodes from 1 to 301 in increments of 5 nodes. An L-OCI estimator is obtained to be used on different test scenarios.

The first test scenario uses the same input set (SF, BW, BCN) as in the training phase and we vary the number of nodes from 5 to 305 in increments of 5 nodes. The L-OCI estimations are shown in figure 5.8 to the left. The RSME obtained in this case is 2.33. The second test scenario uses the parameters ($SF = 12, BW = 125, BCN = 128$), yielding $w = 70$ slots in the frame. We also vary the number of nodes from 5 to 305 in increments of 5 nodes. The L-OCI estimations are shown in figure 5.8 to the right. The RSME obtained is 42.17.



(a) Test scenario with w=140



(b) Test scenario with w=70

Figure 5.8: L-OCI estimator trained with w=140 slots

It is clear from the results that the L-OCI estimator obtained cannot be used in the second test case due to the different naive curve obtained caused by using a different number of slots in the frame. An L-OCI estimator should then only be used on test scenarios having the same number of slots as in the training phase used.

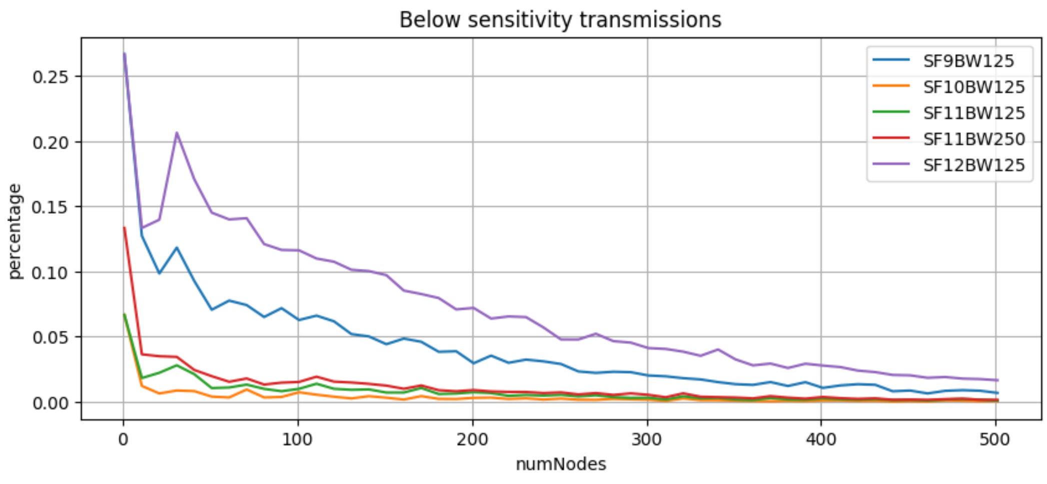
5.2.2. Sensitivity to LoRa parameters

In this experiment we show the impact of using an L-OCI estimator trained on different sets of parameters but all having the same number of slots w=167. We consider several estimation phases using a frame with w slots to obtain different L-OCI estimators to be used on a single test phase. The estimation phase input parameters are shown in table 5.3. We vary the number of nodes from 1 to 501 with increments of 10 nodes for all cases.

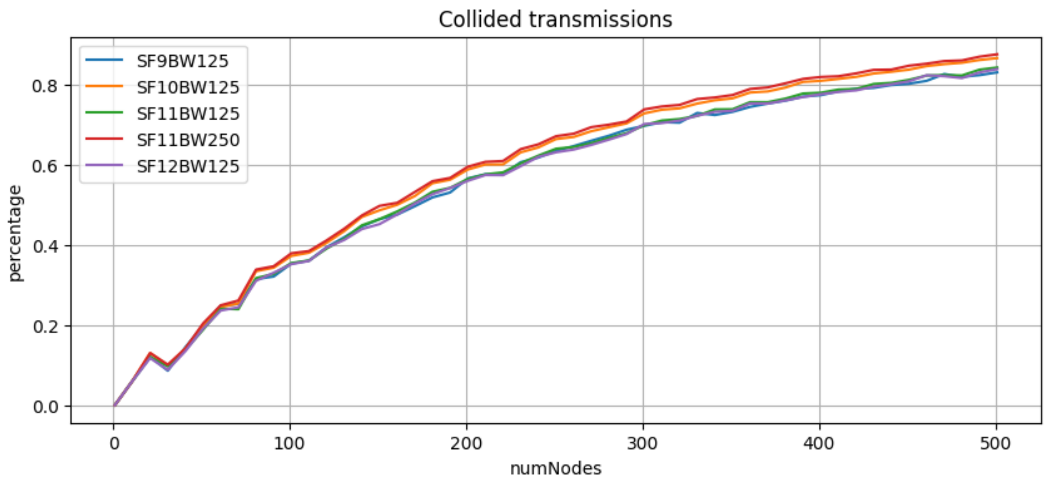
Tabla 5.3: Estimation phase input parameters (w=167) and RSME error achieved during the test phase

SF	**BW**	**BCN**	**RSME**
9	125	50	29.693
10	125	91	5.556
11	250	80	5.586
11	125	152	5.747
12	125	295	17.909

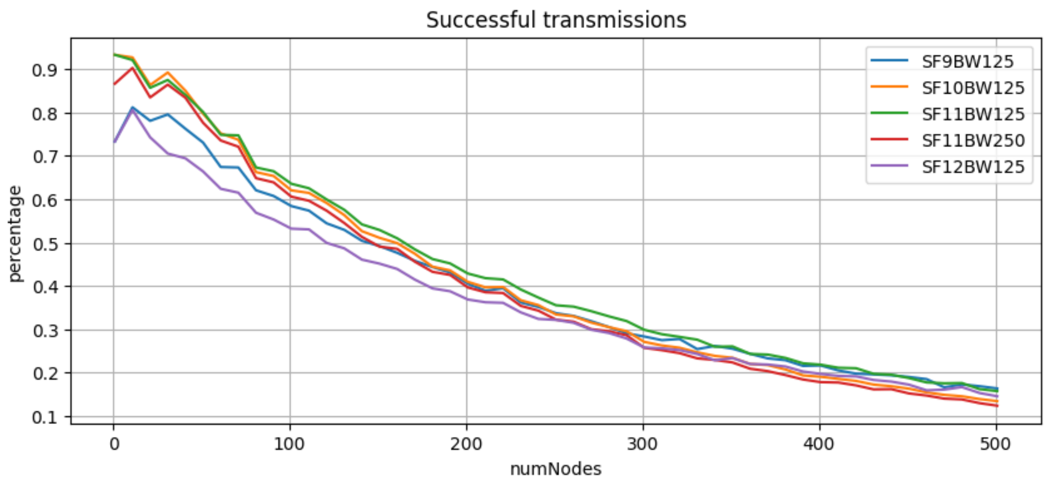
The transmission outcome for the training phase scenarios is shown in figure 5.9. It is noted that the cases with the highest number of lost transmissions due to low power are the ones with SF=9 and SF=12, which correspond to the cases with the shortest range and the longest satellite displacement during the frame, respectively.



(a) Below sensitivity



(b) Collided



(c) Successful

Figure 5.9: Transmissions outcome during estimation phase

The test phase input parameters are $(SF = 10, BW = 125, BCN = 91)$, same as one of the training phase scenarios, and we vary the number of nodes from 5 to 505 with increments of 10 nodes. The L-OCI estimations obtained are shown in figure 5.10 while the RSME error values obtained are shown in table 5.3. The results show the lowest estimation error is obtained when the training and test phase use the same input parameters. However, in other two cases having similar transmission outcomes the error curves obtained are comparable. Therefore, the results indicate that L-OCI estimator could still be used on a scenario with different study parameters but having the same number of nodes and similar communication conditions.

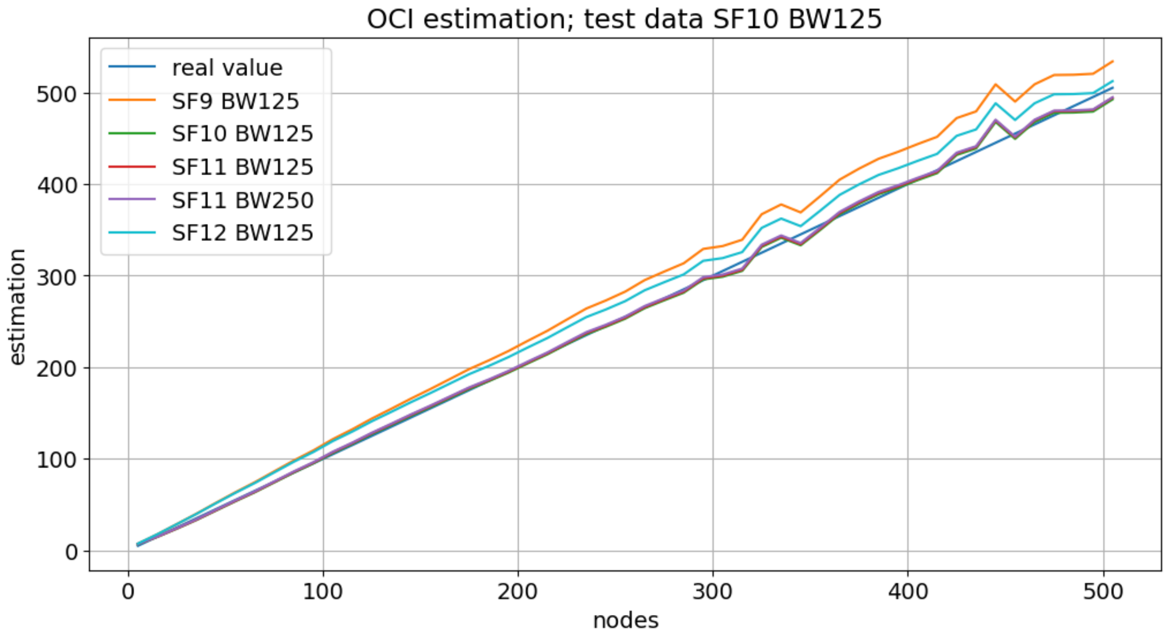


Figure 5.10: Comparison of multiple L-OCI estimators

5.2.3. Impact of L-OCI on the throughput

In this experiment, we show the impact of using the L-OCI estimations to support a MAC protocol. We selected the Slotted ALOHA Game protocol, presented in [9]. The focus of this analysis is on the throughput obtained during the test phase, calculated as the number of successful transmissions in a frame divided by the time length of the frame, and how it reflects the RSME of L-OCI estimations.

Consider the L-OCI estimations obtained from the previous experiment 5.2.2, in particular the ones with SF values 9, 10, and 12, and a BW of 125 kHz. Figure 5.11 shows the throughput obtained during test phase when feeding the L-OCI estimations to the Slotted ALOHA Game MAC protocol. We compare the results with the pure FSA scenario, in which the protocol does not adapt to the estimated size of the network.

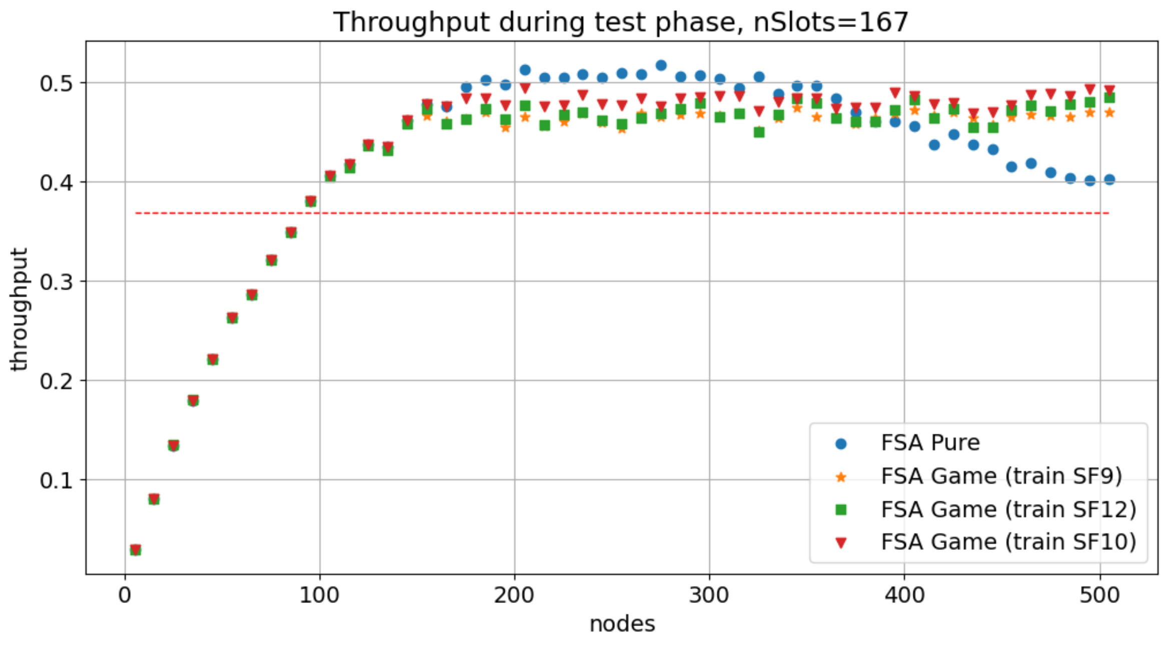


Figure 5.11: Throughput for three scenarios with different L-OCI network size estimation feedback

The first thing to notice is that the theoretical limit of the pure Slotted ALOHA scheme (red horizontal line) is surpassed, given that in FLoRaSat collisions are not destructive and, therefore, the first signal to arrive during a certain slot is decoded successfully but the rest are labeled as collided transmissions.

The throughput of the pure FSA scenario reaches a maximum level between 200 and 300 nodes, higher than in the scenarios with FSA Game MAC control, but it starts decreasing steadily after that point. Using FSA Game, the throughput attains and average maximum value and it is maintained thereafter regardless of the number of nodes. The average throughput values observed after 200 nodes are: 46.5% for the SF9 scenario, 48.1% for SF10, and 46.8% for SF 12. The highest average throughput obtained is when using the L-OCI estimation obtained with SF10 for the FSA Game, which is also the one who achieves the lowest RSME (see table 5.3), however, the difference in throughput with respect to the other 2 scenarios using L-OCI estimations with higher RSME, i.e. SF9 and SF12, is minimal. Moreover, the scenario using the estimations with the highest error (SF9) achieve a slightly better throughput than the scenario using the estimation obtained with SF12.

5.2.4. Error and Energy trade-off

In this experiment, we study the estimation error measured in RSME and how it increases with the number of nodes and the number of slots. We select SF=12 and BW=125 for LoRa while BCN is chosen in function of the number of slots desired. A series of estimation phases and corresponding test phase are performed. For each pair, the number of slots w_i is fixed and the number of nodes is increased up to an arbitrary limit given by the naive estimation. The procedure is the following:

- Train a L-OCI estimator with input parameters $(SF = 12, BW = 125, BCN_i)$. Then vary the number of nodes from 5 to end_i with a step of $step_i$

- For the test phase, fix the input parameters (SF, BW, BCN). Then vary the number of nodes from 3 to $end_i - 2$ with a step of $step_i$
- Calculate the RSME error of the L-OCI estimations obtained. The error is a function of the number of nodes end_i and the number of slots w_i
- Repeat steps 1 to 3 increasing end_i until reaching the limit for the naive estimations so they do not have repeated values
- Repeat steps 1 to 4 changing the number of slots w_i

This procedure allows us to obtain a series of RSME curves in function of the number of nodes, where on each point, the test phase includes up to (almost) the same amount of nodes used for training. The results are shown in figure 5.12

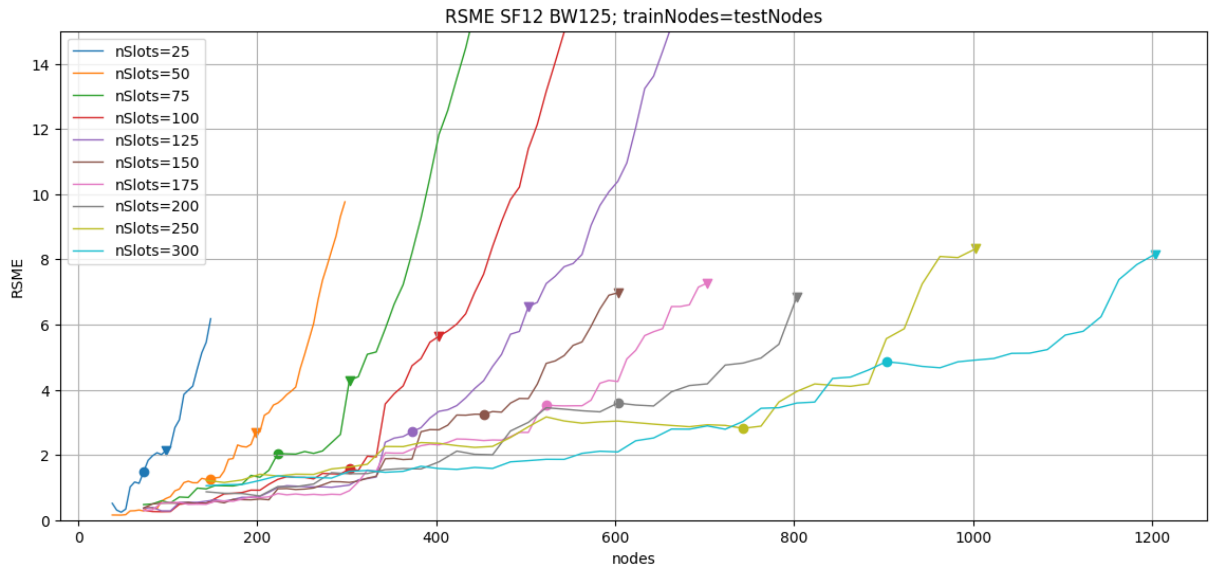


Figure 5.12: RSME for SF12 BW125 and different slot number w

Some interesting discussions come from the results. Consider for instance we aim to estimate a network of up to 600 nodes, so it would be possible to use a frame with w=300 slots to have a low estimation error, or even more slots. However, the more slots a frame has, the longer the frame and the more the satellite moves, increasing the probability of having lost transmissions due to low reception power. Even though L-OCI can still achieve a low estimation error under this scenario, it was demonstrated that the L-OCI estimator obtained is tied to its behavior during estimation phase, so if a low number of successful transmissions is obtained then the results will be replicated during test phase and the estimation obtained may not be as useful to assist a MAC protocol. On the other hand, one could choose to use a frame with w=175, compromising estimation error. This reduces the probability of lost transmission, but, at the same time, increasing the probability of collisions.

The following experiment evaluates the effectiveness of the MOP heuristic proposed in Section 3.2 for optimizing energy consumption and estimation error in LoRa networks. The wasted power and RSME are jointly analyzed in Fig. 5.13, where the former is calculated as the power spent by a node in non-successful transmissions.

A parameter selection campaign was conducted for 100, 500, 1000, and 1500-node network sizes. Results show that wasted energy and estimation error increase with the number of nodes in the network. Moreover, the most energy-efficient solutions were obtained using $SF = 9$ and $BW = 125$ for all cases. For instance, the Pareto curve for a network of 500 nodes suggests that a parameter set with $SF = 9$ and $BW = 125$ can achieve an estimation error of $RSME = 5$ and a wasted energy per node of $16mW$. To further reduce the error, switching to $SF = 11$ and $BW = 250$ is required, resulting in a reduced error of $RSME = 2.5$ but increased wasted energy to $27mW$ per node (see red arrow in Fig. 5.13). The results demonstrate the effectiveness of the MOP heuristic for optimizing energy utilization and estimation error in LoRa networks. Furthermore, in light of the observations in the previous experiment, the marginal improvement in error might not be justified in terms of throughput due to the significant increase in energy consumption.

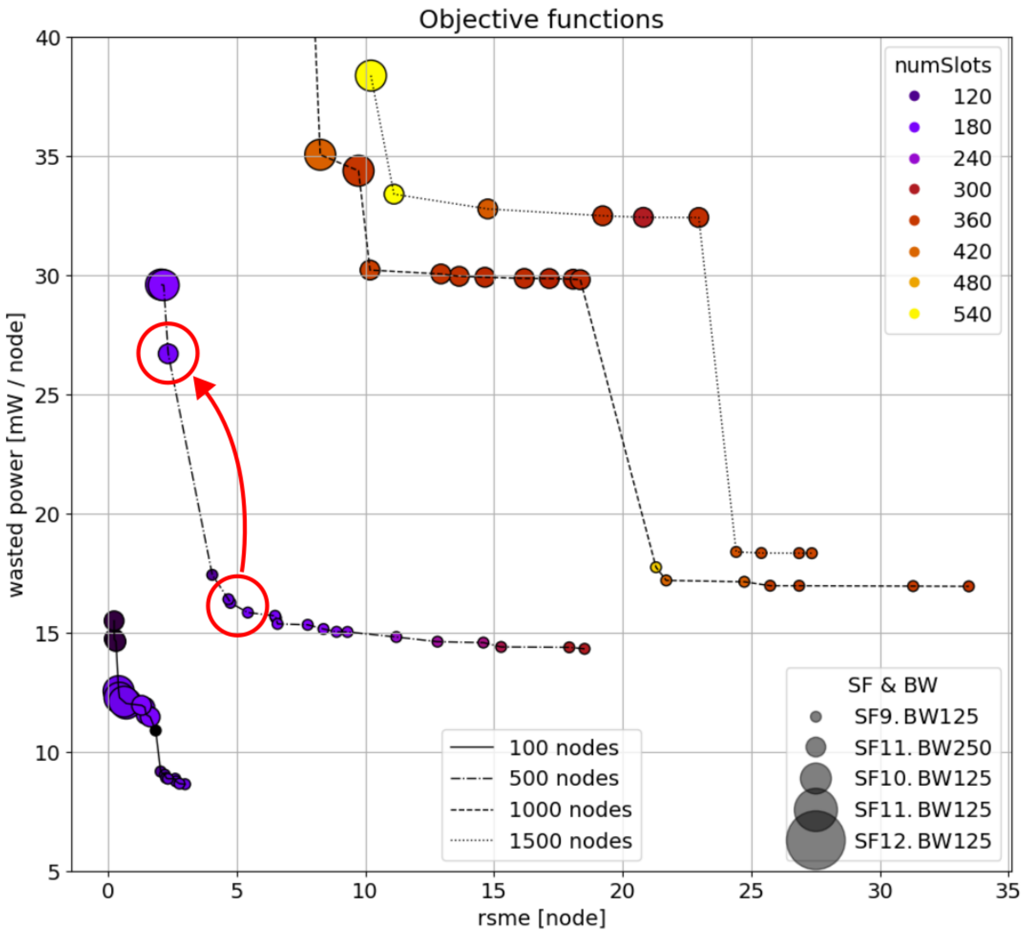


Figure 5.13: Pareto curves for different network sizes obtained from MOP

Chapter 6

Conclusion

In this work, we proposed L-OCI: a LoRa/LoRaWAN realization of the Optimistic Collision Information (OCI) for Direct-to-Satellite IoT (DtS-IoT). We evaluated L-OCI in realistic LoRa-based DtS-IoT networks providing the first evidence of the expected performance of the approach. An extensive simulation campaign showed that L-OCI maintains low error estimations and power efficiency, even in scenarios with a high packet loss rate. Also, L-OCI proved insensitive to $SF = 10$ and $SF = 11$ regarding RSME and throughput, suggesting a single estimator can fit both use cases. However, in resource-full satellites, multiple estimators could co-exist for optimal performance. Furthermore, our analysis showed that $SF = 9$ and $BW = 125kHz$ are appealing candidates for improving energy efficiency if the link budget can be closed under such parameters.

The study found that L-OCI is well-suited for LoRa-based DtS-IoT networks, but the modulation parameters and beacon period also impact communication. Multiple OCI estimators may be required for optimal performance, but when resources are limited, a single estimator can support diverse parameters with minimal error and negligible impact on throughput. SF9 provides optimal energy utilization at the expense of reduced error, but higher SF may be necessary for tighter link budgets. Future work includes an extended analysis with different orbital parameters, more training parameters for L-OCI, and L-OCI extensions considering multiple SFs in the same frame.

Bibliography

[1] Fraire, J. A., Céspedes, S., y Accettura, N., "Direct-to-satellite IoT-a survey of the state of the art and future research perspectives: Backhauling the IoT through LEO satellites," en ADHOC-NOW 2019, Proceedings 18, pp. 241–258, Springer, 2019.

[2] Fraire, J. A., Henn, S., Dovis, F., Garello, R., y Taricco, G., "Sparse satellite constellation design for lora-based direct-to-satellite internet of things," en GLOBECOM 2020-2020 IEEE Global Communications Conference, pp. 1–6, IEEE, 2020.

[3] Gu, F., Niu, J., Jiang, L., Liu, X., y Atiquzzaman, M., "Survey of the low power wide area network technologies," Journal of Network and Computer Applications, vol. 149, p. 102459, 2020.

[4] Fraire, J. A., Iova, O., y Valois, F., "Space-terrestrial integrated internet of things: Challenges and opportunities," IEEE Comms. Magazine, 2022.

[5] Vincent, C., Mcconnell, B. J., Ridoux, V., y Fedak, M. A., "Assessment of argos location accuracy from satellite tags deployed on captive gray seals," Marine Mammal Science, vol. 18, no. 1, pp. 156–166, 2002.

[6] Patmasari, R., Wijayanto, I., Deanto, R., Gautama, Y., y Vidyaningtyas, H., "Design and realization of automatic packet reporting system (APRS) for sending telemetry data in nano satellite communication system," Journal of Measurements, Electronics, Communications, and Systems, vol. 4, no. 1, pp. 1–7, 2018.

[7] Carson-Jackson, J., "Satellite AIS – developing technology or existing capability?," Journal of Navigation, vol. 65, no. 2, p. 303–321, 2012, doi:10.1017/S037346331100066X.

[8] Werner, K., Bredemeyer, J., y Delovski, T., "ADS-B over satellite: Global air traffic surveillance from space," en 2014 Tyrrhenian International Workshop on Digital Communications-Enhanced Surveillance of Aircraft and Vehicles (TIWDC/ESAV), pp. 47–52, IEEE, 2014.

[9] Zhao, B., Ren, G., y Zhang, H., "Slotted aloha game for medium access control in satellite networks," en 2019 IEEE/CIC International Conference on Communications in China (ICCC), pp. 518–522, IEEE, 2019.

[10] Ilabaca Parra, P., Montejo-Sánchez, S., Fraire, J. A., Souza, R. D., y Céspedes, S., "Network size estimation for direct-to-satellite iot," IEEE Internet of Things Journal, vol. 10, no. 7, pp. 6111–6125, 2022.

[11] Sornin, N., Luis, M., Eirich, T., Kramp, T., y Hersent, O., "Lorawan specification," LoRa alliance, vol. 1, 2015.

[12] Fraire, J. A., Madoery, P., Mesbah, M. A., Iova, O., y Valois, F., "Simulating lora-based direct-to-satellite iot networks with florasat," en 2022 IEEE 23rd WoWMoM,

pp. 464–470, IEEE, 2022.

[13] LoRa_Alliance, “Lorawan specification v1.1,” 2017, https://lora-alliance.org/resource_hub/lorawan-specification-v1-1/.

[14] Chen, P.-Y., Bhatia, L., Kolcun, R., Boyle, D., y McCann, J. A., “Contact-aware opportunistic data forwarding in disconnected lorawan mobile networks,” en 2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS), pp. 574–583, IEEE, 2020.

[15] Abramson, N., “The aloha system: Another alternative for computer communications,” en Proceedings of the November 17-19, 1970, fall joint computer conference, pp. 281–285, 1970.

[16] Chasserat, L., Accettura, N., y Berthou, P., “Short: Achieving energy efficiency in dense lorawans through tdma,” en 2020 IEEE 21st International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM), pp. 26–29, IEEE, 2020.

[17] Chasserat, L., Accettura, N., y Berthou, P., “Lorasync: energy efficient synchronization for scalable lorawan,” 2022.

[18] Vallado, D. y Crawford, P., “Sgp4 orbit determination,” en AIAA/AAS Astrodynamics Specialist Conference and Exhibit, p. 6770, 2008.

[19] “Cubesat design specification.” https://static1.squarespace.com/static/5418c831e4b0fa4ecac1bacd/t/5f24997b6deea10cc52bb016/1596234122437/CDS+REV14+2020-07-31+DRAFT.pdf. Accessed: 2023-04-13.

[20] Gonzalez, C., Rojas, C., Becerra, A., Rojas, J., Opazo, T., y Diaz, M., “Lessons learned from building the first chilean nano-satellite: The suchai project,” 2018.

[21] Han, C., Zhang, Y., Bai, S., Sun, X., y Wang, X., “Novel method to calculate satellite visibility for an arbitrary sensor field,” Aerospace Science and Technology, vol. 112, p. 106668, 2021.

[22] “Omnet++ framework.”, https://omnetpp.org/.

[23] Varga, A. y Hornig, R., “An overview of the omnet++ simulation environment,” en 1st International ICST Conference on Simulation Tools and Techniques for Communications, Networks and Systems, 2010.

[24] “Inet framework.”, https://inet.omnetpp.org/.

[25] Slabicki, M., Premsankar, G., y Di Francesco, M., “Adaptive configuration of lora networks for dense iot deployments,” en NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium, pp. 1–9, IEEE, 2018.

[26] Niehoefer, B., Šubik, S., y Wietfeld, C., “The cni open source satellite simulator based on omnet++,” en Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques, pp. 314–321, 2013.

[27] Valentine, A. y Parisis, G., “Developing and experimenting with leo satellite constellations in omnet++,” arXiv preprint arXiv:2109.12046, 2021.

[28] Boquet, G., Tuset-Peiró, P., Adelantado, F., Watteyne, T., y Vilajosana, X., “Lr-fhss: Overview and performance analysis,” IEEE Communications Magazine, vol. 59, no. 3, pp. 30–36, 2021.

[29] "Two-line element." https://celestrak.org/NORAD/documentation/tle-fmt.php. Accessed: 2023-04-13.

[30] LoRa_Alliance, "Rp2-1.0.2 lorawan® regional parameters," 2020, https://lora-alliance.org/resource_hub/rp2-102-lorawan-regional-parameters/.

[31] SEMTECH, "Semtech sx1272/73 lora transceiver documentation," 2019, https://www.semtech.com/products/wireless-rf/lora-connect/sx1272#documentation.