A Survey of LoRaWAN Simulation Tools in ns-3

Jéssika C. da Silva, Daniel de L. Flor, Vicente A. de Sousa Jr., Níbia Souza Bezerra, Alvaro A. M. de Medeiros

***Abstract*—The Internet of Things (IoT) paradigm gains more importance every day, as the number and variety of connected devices grows. To answer the specific needs of IoT, many wireless network standards have been developed. An example is the LoRa Wide Area Network (LoRaWAN), which has gained popularity thanks to the affordable costs of its devices and gateways, as well as wide range, low energy consumption and flexibility. Considering the challenges that exist with testing and researching on real LoRaWAN systems, the development of accurate network simulators is invaluable. Therefore, this work presents a survey of the available tools for simulating LoRa networks in ns-3 network simulator. We present how those simulators were implemented and their main features and limitations. We also showcase how those simulators are being used in the literature to evaluate the performance of LoRA networks. Finally, we compare the modules to highlight what scenarios each of them is more suited for.**

***Index Terms*—LoRa, LoRaWAN, ns-3, LPWAN, IoT, network simulators.**

I. Introduction

The IoT, a paradigm in which an enormous number of devices can be interconnected, has drawn the attention of many researchers and investors through the years. The IoT is set to have a significant impact on our lives, as more and more devices are added to a smart network capable of connecting everything we want. Applications exist in a wide range of scenarios, from smart buildings to smart cities, enabling important changes on the production and manufacture process in the form of the Industry 4.0 [1].

The importance of IoT for today and for future networks is evidenced by the technical requirements of the International Mobile Telecommunications-2020 (IMT-2020) systems, the next generation (5G) of mobile services. The International Telecommunications Union (ITU), the organization responsible for creating standards and recommendations to enable world-wide information and telecommunications systems, has stated through its technical reports [2] that the evolution of mobile systems must go beyond the increase in data rates. 5G is expected to work not only in smartphones, but in smart things. The report in [2] specifies three major scenarios for 5G networks: mobile broadband, mission-critical communication, and massive-machine communications.

Jéssika C. da Silva, Daniel de L. Flor and Vicente A. Sousa Jr. are with the Federal University of Rio Grande do Norte, Natal, RN, Brazil (e-mails: {jessie,danielflor,vicente.sousa}@ufrn.edu.br).

Níbia Souza Bezerra is with the Luleå University of Technology, Skellefteå, Sweden (e-mails: nibia.souza.bezerra@ltu.se).

Alvaro A. M. de Medeiros is with Federal University of Juiz de Fora (e-mail: alvaro@engenharia.ufjf.br.).

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. The proof of concept simulations provided by this paper was supported by High Performance Computing Center (NPAD/UFRN).

Digital Object Identifier: 10.14209/jcis.2021.2

The latter is known as Massive Machine-Type Communications (mMTC). This scenario requires a system to support a connection density of one million devices per km^2, with relaxed data rates and latency demands. Energy efficiency is also of concern, as mMTC-enabled devices should have a battery life of 10 to 15 years. These requirements show a great overlap between some IoT applications and mMTC systems. Given how ubiquitous massive machine communications are becoming, it is important to understand what are the current technologies that could support those scenarios, and what tools are available to evaluate them.

A. mMTC Enablers

The specific requirements of mMTC systems impose technical challenges on the Physical (PHY) and Medium Access Control (MAC) layers. There are different standards and protocols that can fulfill some of those requirements. However, there is not yet a single technology capable of fully supporting the wide range of IoT applications. In general, as shown in Table I, the enabling wireless communication technologies can be grouped in three types:

- *Low-Rate Wireless Personal Area Networks (LR-WPANs)* have a short range and low data rates, reducing power consumption of devices. LR-WPANs are focused on applications that connect a relatively small amount of devices in a small area;
- *Low-Power Wide-Area Networks (LPWANs)* are characterized by wide area coverage. Since they were conceived with IoT applications in mind, the design choices in LPWANs take into account the battery restraints.
- *Cellular-based networks* are LPWANs that leverage the infrastructure of the already well-established mobile systems, such as Global System for Mobile Communications (GSM) and Long Term Evolution (LTE). They are adapted versions of the traditional cellular networks, in order to support massive connection and low power requirements;

TABLE I
Different types enabling Wireless Technologies.

Type	Key Aspects	Examples
LR-WPAN	Low data rates and range, small number of devices, established technology	Bluetooth, Zigbee, 6LoWPAN
LPWAN (cellular-based)	Uses mobile network infrastructure, high cost to implement	LTE-M, NB-IoT, EC-GSM-IoT
LPWAN	Long range, low power consumption	SigFox, LoRaWaN, Weightless

The main competitors in the LPWAN category are Sigfox and LoRaWAN, both based on proprietary technologies. Sigfox utilizes Ultra Narrow Band (UNB) technique, and allows for very small messages (up to 12 bytes in size) using very low bandwidth (100 Hz) and a low data rate (100 bps). These configurations, along with the low signaling overhead of the protocol, allow for a wide coverage with little energy consumption [3].

LoRaWAN is a standard based on the LoRa radio technology. LoRa uses a spread spectrum technique to enable a receiver with high sensitivity, allowing for a trade-off between range and data rate. Most of the control functions of the protocol are located in central nodes, thus enabling the devices to be cheaper and simpler. This also helps with reducing energy consumption. LoRa and LoRaWAN have gained a lot of attention recently. Many studies assessing its capabilities and limitations have been published [4]–[7]. Thus, with the purpose of attaining a survey on prototyping tools for IoT features, we adopt LoRaWAN as our target enabling technology.

B. Network Simulation Tools

Network simulators are very important tools for network research and development. They allow us to study scenarios that are difficult or expensive to investigate in real systems or to obtain data from in a controlled and reproducible way. A particular example is the study of LPWANs, which may have thousands of devices connected in a single network. Table II presents five of the most popular simulators for wireless networks. All five simulators have support for different wireless standards (such as the 802.11 family) and models for traffic generation, mobility, channel propagation and energy consumption. Table II also shows the programming languages used to build each simulator, their type of license (open-source or proprietary) and if they have or not a Native GUI Support. Of the five tools presented, only ns-2, ns-3 and OMNeT++ are open-source. More information on these and other simulators can be found in [8].

Several LoRaWAN tools have been developed based on these discrete-event network simulators, and others were developed specifically for LoRaWAN networks. Table III presents the main publicly available simulators, along with the code language used and links to their website.

Despite the OMNet+ and LoRaSIM environments being accessible alternatives, this work focus on the ns-3 simulator options due to a number of reasons. LoRaSIM is a simulator developed only for LoRaWAN networks, which limits the tools available for creating simulations campaigns and exploring different aspects of the network. OMNet+ has incomplete support for wired and wireless network simulation, and it requires significant background work, since very few protocols have been implemented [8]. On the other hand, the ns-3 is the most flexible choice [8]. It has continuous support and documentation, and includes implementations for many protocols and standards. Due to these advantages, many ns-3-based tools for LoRaWAN were developed, as shown in Table III.

The ns-3 is a discrete-event network simulator primarily used for research and educational purposes [15]. The simulator's main code is written in C++, with Python being used for bindings. In its core there is a set of libraries (models) that can simulate many aspects of a packet-based network, such as:

- Traffic, packet error, channel propagation and mobility models;
- Detailed PHY/MAC layers of systems like LTE and Wi-Fi, following their respective standards;
- Upper-MAC layer protocols such as IPv6 and TCP, emulating a real Ethernet protocol stack.

Many researchers have developed LoRaWAN network-level simulation tools based on ns-3, in order to evaluate the performance of the technology in several scenarios. This work aims to be a reference for a first comparison between ns-3 LoRaWAN modules for ns-3 users. It also provides relevant information about the LoRaWAN standard and on the use of the modules, allowing other researchers to easily and quickly start their own studies with the simulators.

C. Related Work

Many surveys or review papers about LoRa are available in the literature. The survey [16] presents a complete literature review and analysis of the research published from 2015 to September 2018 that are accessible via Google Scholar and IEEE Explore databases. It covers a wide range of papers and categorizes them in aspects like designing network-level simulators, deployed LoRaWAN testbeds, application studies, LoRaWAN improvements, mathematical and empirical LoRaWAN network models. Additionally, it brings an overview of LoRa and LoRaWAN technologies, discusses challenges that still need to be addressed in LoRaWAN, and compile all the knowledge considered providing a Strength, Weakness, Opportunity, and Threat (SWOT) analysis of LoRaWAN. The work in [17] has a similar purpose as the work in [16], although there is a more compact literature review.

Paper [18] is an overview paper that brings analysis and discussions about the performance evaluation done via a test-bed and simulations. The paper [19] contextualizes the LoRaWAN protocol on the IoT scenario highlighting obstacles, challenges, open issues, use cases, and research opportunities, with a brief comparison with other standard protocols. Similarly, the survey [20] validates LoRa technology as a good candidate to fulfill IoT requirements.

Although the survey [16] has brief comments about several LoRaWAN system-level simulators on one of their categories, some of the existing simulators are missing. No work until now has tested and analyzed all the different implementations of LoRaWAN modules in ns-3. In this paper, we present a survey of these simulators, along with their key features and limitations. We labeled them as *Module I* through *IV*, in order of the date they were made publicly available. Most of these modules were already extended with additional LoRaWAN features and improvements after their first release, which are also presented in this article.

TABLE II
POPULAR SIMULATION TOOLS FOR WIRELESS NETWORKS.

Simulator	Language	License	Native GUI Support	URL
ns-2	C++ and OTcl	Open source	No	http://nsnam.sourceforge.net/wiki/index.php/Main_Page
ns-3	C++ and Pyhton	Open source	No	https://www.nsnam.org
OMNeT++	C++	Open Source	Yes	https://omnetpp.org
OPNET	C and C++	Proprietary	Yes	https://www.riverbed.com/products/steelcentral/opnet.html
NetSim	C and Java	Proprietary	Yes	https://www.tetcos.com

TABLE III
POPULAR SIMULATION TOOLS FOR LORAWAN.

Simulator	Language	URL
Module I - ns-3 [9]	C++ and python	https://github.com/signetlabdei/lorawan
Module II - ns-3 [10]	C++ and python	https://github.com/networkedsystems/lora-ns3
Module III - ns-3 [11]	C++ and Python	https://github.com/imec-idlab/ns-3-dev-git/tree/lorawan
Module IV - ns-3 [12]	C++ and python	https://github.com/drakkar-lig/lora-ns3-module
FLoRa - OMNeT++ [13]	C++	https://flora.aalto.fi/
LoRaSim - SimPhy [14]	Python	https://www.lancaster.ac.uk/scc/siteelcentral/opnets/lora/lorasim.html

The remainder of this work is organized as follows. Section II briefly describes the major technical details of LoRa and LoRaWAN. Section III presents our survey of the available modules for ns-3. Section IV presents our experiences when testing the modules. Finally, Section V concludes with our comments on the different implementations.

II. LORA AND LORAWAN TECHNOLOGY

The term LoRa (meaning Long Range) is commonly used to refer to three distinct aspects of the technology. The first and correct aspect is the modulation scheme employed by the transceivers. In this case, it refers to the proprietary PHY-layer technology developed by French company Cycleo, which was acquired in 2012 by Semtech Corporation, the sole supplier of LoRa-enabled Radio Frequency (RF) chips. LoRa can also be wrongly used to refer to two other aspects: the network of end-devices and gateways that employ LoRa modulation, or to the MAC-layer specification of these networks. Both of these are actually related to the LoRaWAN standard.

The LoRaWAN standard is maintained by the LoRa Alliance, a group of research institutions and companies that are interested in the development and expansion of LoRa-based LPWAN IoT networks. Despite being based on a patented PHY implementation, the LoRaWAN standard is open-source and described in [21]. It defines physical layer parameters, frame formats, classes of devices and security and network protocols.

The remainder of this section examines the key aspects of the LoRa PHY-layer and LoRaWAN MAC-layer specifications.

A. LoRa Physical Layer

The LoRa modulation is based on a spread spectrum technique called Chirp Spread Spectrum (CSS). Like the Direct Sequence Spread Spectrum (DSSS) technique, CSS spreads a signal across a bandwidth wider than the minimum required for its data rate. Thus, the resulting transmission is robust against narrowband noise. This characteristic allows LoRa to improve the receiver's sensitivity and extend the range of communication, at the cost of a lower spectral efficiency when compared to narrowband transmissions [22].

In DSSS, a signal is spread by means of an operation with a pseudo-random sequence. A bit on the original signal is represented by a certain number of "chips". Since the chip rate is higher then original signal's rate, the resulting spectrum is wider.

In contrast, CSS utilizes a carrier signal whose frequency varies continuously over the entire bandwidth to obtain the spread effect. This signal is called a *chirp*.

Each chirp transmitted corresponds to a single symbol, thus the symbol period is equal to the chirp duration, T_s.

The number of possible starting frequencies and therefore the number of symbols a chirp can code depends on the Spreading Factor. The Spreading Factor (SF) is a parameter of the LoRa modulation, and is defined by the relation:

$$2^{SF} = BW \cdot T_s, \tag{1}$$

where SF also determines the number of code words as 2^{SF}. Therefore, each chirp can carry SF bits, and the interval Δf is

$$\Delta f = \frac{BW}{2^{SF}}. \tag{2}$$

The spreading factor can vary from 7 to 12. It is also related to the chirp duration. This is because the chip rate R_c, defined as

$$R_c = \frac{2^{SF}}{T_s} = BW, \tag{3}$$

is always constant and equal to the bandwidth BW. Therefore, the chirp duration doubles with each increment of the SF. This results in the expression for the raw bit rate R_b, which is

$$R_b = \frac{SF}{T_s} = SF\frac{BW}{2^{SF}}. \tag{4}$$

Table IV shows different data rates computed using Equation (4). The bandwidth in LoRa transceivers can be set to 125, 250 or 500 kHz. Other expressions and details on the

LoRa modulation can be found in a document provided by Semtech in [23].

TABLE IV
DATA RATES (IN KBPS) FOR DIFFERENT COMBINATIONS OF BW AND SF (BEFORE FEC IS APPLIED).

SF ↓	125 kHz	250 kHz	500 kHz
7	6.835	13.671	27.343
8	3.906	7.812	15.625
9	2.197	4.396	8.793
10	1.220	2.441	4.8823
11	0.671	1.342	2.685
12	0.366	0.732	1.464

Increasing SF allows more bits per symbol, as well as a chirp of longer duration, meaning that the signal is more robust to interference or noise. This, however, lowers the data rate of the communication, as seen in Equation 4.

At reception, demodulation is performed my multiplying the signal with an un-shifted down-chirp signal of same BW and SF. The resulting signal is sampled at the chip rate, i.e., BW Hz. Then, the Fast Fourier Transform (FFT) is computed, creating a signal with a peak shifted by the shifting value used to encode the symbol. More details about the demodulation process can be found in [24]. Since the Lora Technology is patented and there is not a publicly available description of the signal processing equations, the paper [25] gives the first rigorous mathematical signal processing description of the modulation and demodulation processes and a theoretical derivation of the optimum receiver.

In addition to multistate CSS, LoRa employs some encoding schemes before modulation and transmission to further improve its robustness and reliability [26]. Data whitening, Forward Error Correction (FEC), Interleaving and Grey Mapping are applied to the data before modulation and transmission. By adjusting its transmission power, SF, bandwidth and FEC coding rate, LoRa is able to adapt to different conditions to ensure a good link quality, or to optimize energy consumption and data rate. For example, if the battery life of the device is critical, SF and BW can be set in their lower values to minimize the time on air of packets and noise degradation over the bandwidth.

B. LoRaWAN MAC Layer

The architecture of a LoRaWAN is represented in the Fig.1. It follows a star-of-stars topology. The *End-Devices (EDs)*, typically sensors and actuators, can only send/receive messages to/from *Gateways (GWs)*, using a LoRa link. The gateways forward EDs messages to the *Network Server (NS)* via a high capacity link (like a cellular, Ethernet or optical link). The NS can process the information received from the ED and run a customer's application locally or remotely. It can also send downlink messages to the EDs via the gateways, if necessary.

As shown in the Fig. 1, an ED is not explicitly paired with a single GW. It simply sends its packets to any available gateway in its range, which then receives and forwards these

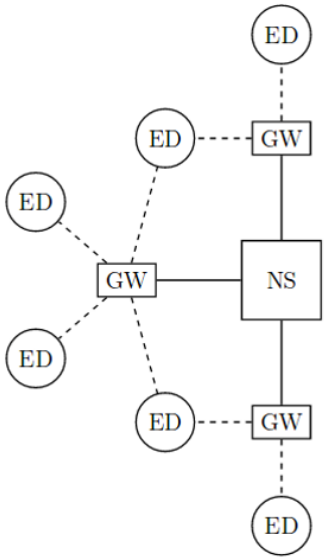

Fig. 1. Basic topology of a LoRaWAN. Reproduced from [26].

messages to the NS. It is the server's responsibility to filter duplicate packets and decode them. The NS also assigns the best gateway to deliver a packet to an ED. The NS and ED are logically linked and the GW is just a bidirectional relay, transparent to the application running in the ED and NS.

The LoRaWAN standard specifies several MAC commands that the NS can use to exchange information and control the behavior of an ED [18], [21]. The NS can, for example:

- Request reports on the battery life or configuration of EDs (*DevStatusReq*);
- Report to EDs on their link quality by replying the received power at the GW (*LinkCheckReq*);
- Issues limitations to the ED's transmit time to regulate network traffic or to comply with Duty Cycle regulations (*DutyCycleReq*);
- Set the transmission parameters of an ED, thus enabling *Adaptive Data Rate (ADR)* algorithms (*LinkADRReq*);

LoRaWAN also specifies three classes of end-devices. Each class has different behaviors and features in their MAC layer:

- **Class A** is the basic class. All LoRaWAN EDs are required to support its features. Class A devices allow asynchronous, bi-directional transmissions, initiated by the ED. A packet is scheduled to be transmitted according to the device's needs, on a random basis (an ALOHA-like protocol). Two short downlink windows are opened after an uplink transmission, allowing the NS to send data or commands. The windows are opened on different frequency channels previously agreed by the NS and ED. Class A is intended for devices that require low power consumption and do not need to receive downlink messages often;
- **Class B** devices can open scheduled downlink slots in addition to the random Class A receive windows. To do so, Class B devices can synchronize with the NS using beacon signals periodically broadcasted by Class B-enabled gateways. It is intended for devices that need

to receive downlink messages or commands in some frequency from an application running remotely;
- **Class C** devices are always available to receive downlink packets, if they are not transmitting. Therefore, it is intended for devices that do not have energy constraints and need short latency in their communication.

It is worth mentioning that LoRaWAN also supports Acknowledge (ACK) messages, allowing the NS to confirm a packet's arrival to an ED, and vice-versa. However, this feature is not mandatory, so as to allow end-devices to be as simple as possible. Additionally, despite its ALOHA-like behavior, studies [12], [26] show that the packet loss rate in LoRaWAN networks for each spreading factor is lower than that expected of pure ALOHA. This is due to the capture effect, since one of two packets sent at the same time on the same SF and band can be correctly received when there is a minimum power difference. Another factor that influences LoRaWAN's packet loss behavior is that transmissions in different spreading factors are not perfectly orthogonal, as showed in [22]. However, most of the simulators wrongly consider that signals sent in different spreading factors never interfere with each other.

LoRaWAN also defines several regional parameters that comply with different regulatory standard for the Industrial, Scientific, and Medical (ISM) bands worldwide [27]. Aspects like frequency bands, channel planning, maximum transmission power, duty cycle, preamble formats and payload sizes are specified. In Europe, for example, LoRaWAN networks are expected to operate with a bandwidth of 125 kHz, on the 433.175 - 433.575 MHz or 868 - 870 MHz range, and with a maximum duty cycle of 1%.

Finally, the standard specifies many other aspects of the protocol such as frame and header formats for the PHY and MAC messages, device activation and join procedures, encryption schemes and more. LoRaWAN also allows non-standard messages formats that are used exclusively within a private network.

C. The ADR Mechanism

The transmission adaptation mechanisms (ADR) enable better leverage of the LoRa PHY layer advantages. It is possible to drastically increase the network capacity when smartly adapting the SF and transmission power in each ED.

The LoRaWAN ADR standard is divided in two parts: one occurs in ED and the other in the NS. The first happens when the connection to the server is lost. In this case, the transmission power value, followed by the SF, are increased until the connection is reestablished. On the NS side, the ADR algorithm receives the SNR and RSSI value of the last transmissions of each device. With these values, the transmission parameters values for each ED can be set by some method of decision. When ADR is activated, the server sends the SF and transmission power chosen values for each ED through MAC commands.

The adaption algorithm greatly influences the network efficiency and the capacity. The Semtech Corporation provides a sample ADR algorithm, although it is not an optimal solution [28]. In this method, the NS keeps in memory the SNR values of the last 20 frames received from each ED. The transmission power and SF values are chosen in order to establish a pre-configured margin between the SNR of the future transmitted packets (with the new power and SF values) and the receiver's minimum sensitivity [28]. With this method, coverage is guaranteed with energy efficiency and adequate data rates for each ED [28].

III. LoRa and LoRaWAN Prototyping in ns-3

We now present a survey of the implementations of LoRaWAN in ns-3. These are mostly available as modules that can be added to an ordinary ns-3 installation. As previously mentioned, we labeled them as *Module I* through *IV*, in order of date they became publicly available. At the end of this Section, Table V presents an overview of the features discussed for each module.

While Module I was tested in ns-3 version 3.27 and 3.29, the other ones were tested in ns-3 version 3.28. It is also worth mentioning that despite the paper [29] adding class B devices to Module III, all four modules originally implement only Class A devices, since it is the mandatory class on LoRaWAN's protocol.

Another important general information is that only modules I, II and III have a LoRaWAN power consumption model implemented, which are based on a simple state machine built only with data provided by LoRa's datasheet. The authors in [30] used real data analyses of energy consumption from the most common LoRa transmitter (SX1272)) together with data provided by LoRa's datasheet for the implementation of a new LoRaWAN energy consumption model in ns-3, which was validated by comparison with an analytical model [6]. They integrated it with modules I, II and III and made them publicly available in [31], [32], [33].

A. Module I

Module I was developed in 2016 as a part of the Masters Thesis of D. Magrin [26], in The University of Padova, Italy. It is available for download in [34]. The authors developed their simulator by simplifying the transmission chain in two main models. A very detailed description of those models is provided in [26]. The first, a *link measurement model*, is used to compute the effects of propagation that affect signal strength at reception. It takes into account the path loss and building penetration loss. It also features a Correlated Shadow Model that considers the correlation between the shadowing experienced by devices that are physically close.

The received power computed by the link measurement model is used by the *link performance model* to evaluate if a transmission was successful and, if otherwise, what caused the packet loss (e.g. low Signal-to-Interference-plus-Noise Ratio (SINR) at the reception). To accomplish this, the model considers the receivers sensitivity (based on the datasheet of the SX1301 chip [35], employed in gateways) and the interference introduced by overlapping packet arrivals, considering the non-orthogonality between symbols of different spreading factors and the capture effect. The

authors choose to use the SINR threshold matrix T presented in [22] to decide whether a packet can survive interference from other LoRa transmissions or not, which is given by

$$T = \begin{bmatrix} 6 & -16 & -18 & -19 & -19 & -20 \\ -24 & 6 & -20 & -22 & -22 & -22 \\ -27 & -27 & 6 & -23 & -25 & -25 \\ -30 & -30 & -30 & 6 & -26 & -28 \\ -33 & -33 & -33 & -33 & 6 & -29 \\ -36 & -36 & -36 & -36 & -36 & 6 \end{bmatrix}. \quad (5)$$

Each element T_{ij} of the matrix has the minimum value of SINR margin required for a packet of SF i to not collide with an interferer packet of SF j, where i and j represent the row and column of the matrix respectively, and start from the lowest SF value (SF 7) to the highest (SF 12).

The matrix main diagonal in Equation (5) represents the capture effect. For instance, one of two overlapping packets at the same SF and frequency can be received correctly if one packet is received at least 6 dB above. The rest of the matrix shows the received power difference between two overlapping LoRa packets of different SFs, in order to be considered orthogonal. Therefore, the higher is the packet's SF, the more resistant it is to the interference of other LoRa signals. The matrix elements were calculated considering complete overlapping between signals. To compensate that, the authors estimate the equalized interfering signal taking into consideration the real overlapping time. One important result of this is that two out of the two overlapping packets with the same SF and similar received power can be correctly decoded when the overlapping time is sufficiently small, so that the equalized SINR is above 6 dB. This fact was actually observed in practical LoRa experiments [36].

Many of these models were implemented by leveraging the already established models in ns-3. For instance, the Propagation Loss Model is implemented using the default *LogDistancePropagationLossModel* class in ns-3. The code is well documented, and a file describing the module is provided. All of this makes it easier to understand and to integrate the code with other ns-3 modules. Finally, the module does not alter existing ns-3 code and can be added to any installation.

One of the main drawbacks of this module is the inability to account for interference of other systems in the ISM band and the simple implementation of the NS, which has a direct link to the gateways. Although the simple NS does not implement all MAC commands, it already has all the structure to allow the implementation of additional MAC commands. The implementation of *LinkADRReq* is the only completed one in the most recent version of the module. The NS also handles the header formats, the addressing of EDs, the logical channel, and the duty cycle management, like the other modules.

Downlink transmissions were absent in the first release of Module I [37], but that feature was later added to the module by its authors. The improved Module I is publicly available in [34], which also was updated with a LoRaWAN energy model and ADR algorithm.

The authors used Module I in [9], where they evaluate throughput and packet loss in scenarios with a single, central gateway. They also analyzed the effects of duty cycle limitations and SF distribution in the network performance. The improved version of Module I, where the authors implemented ACK messages, was used in [38] to study the impact of confirmed traffic in LoRa networks, concluding that it can negatively impact the system performance if not restricted. For the analysis in [39], they improved their module to make it more configurable and appropriated to test network configurations different from the standard. They have obtained significant performance gains when properly setting their proposed configuration parameters, like change in transmission/reception priority, number of parallel reception paths, number of allowed retransmissions, and others.

Other researchers have also used the module. In [40], it was proposed a Multi-User ACK-Aggregation Method. They perform simulations using Module I and showed that the method could improve throughput from 10 % to 30%. In [41], ns-3 was used to validate the proposed analytical mathematical model, which considers SF orthogonality and the capture effect included in downlink and uplink transmissions.

In [42], the authors used ns-3 to compare the capacity of LoRaWAN with IEEE 802.11ah, another LPWAN standard. The result shows that the former performs better regarding this aspect. In [43], the authors evaluated the improvement in the scalability of LoRa networks when persistent-Carrier Sense Multiple Access (p-CSMA), a contention-based medium access strategy, is used as the MAC mechanism instead of the ALOHA-like LoRaWAN standard.

In [44], the module is used to assess the behavior of LoRaWAN in a typical industrial monitoring scenario. The authors used a different model of interference and implemented energy consumption models that interface with the *energy* framework in ns-3. However, they did not make them available. An energy consumption model was also added for the studies in [45], which is based on the existing ns-3 Wi-Fi energy consumption model and LoRa energy specification, but also not publicly available. The authors tested different configurations of SF distribution for network slicing. Two proposals are made: an adaptive dynamic inter-slicing and an intra-slicing resource allocation algorithms. Resources are distributed between slices based on urgency and reliability, and on each slice, resource allocation is optimized based on QoS features.

B. Module II

The Module II was developed by Brecht Reynders and other researchers from KU Keven University, in Belgium. It is available for download at [46]. The authors stated in [10] that their module has been in use for two years in LoRaWAN research.

The module is coded in a similar style to ns-3 default modules. It connects to ns-3 callbacks frameworks, and features "helpers" objects to configure the network. For this reason, despite the lack of documentation and few comments, it is possible to understand the code. The module was submitted to The Workshop on ns-3 (WNS3, 2018) and was accepted for publication [10].

The physical layer is implemented in the *Loraphy* and *Loragwphy*, for ED and GW, respectively. The class *spectrumsignalparameters* is used to set up parameters like SF and bandwidth. The PHY implementation does not take into consideration SINR thresholds due to non-orthogonality between transmissions of different SF as Module I. Still, the interference caused by these transmissions is taken into account, modeled as noise when calculating Signal-to-Noise Ratio (SNR). The probability of bit error (P_e) is given by

$$P_e = \frac{1}{2}Q\left(1.28 * \left(k\frac{SNR * B}{R_B(S)}\right)^{1/2} - 1.28 * k^{1/2} + 0.4\right), \quad (6)$$

where k means the number of bits per symbol, $R_B(S)$ is the bit rate, B is the bandwidth, and $Q(\cdot)$ is the tail distribution function of the standard normal distribution. Equation (6) was obtained through LoRa physical layer simulations in Matlab [10]. An interesting observation of these simulations is the demodulation process of LoRa symbols, which occurred through correlation and not by FFT.

Module II assumes that a packet was not received correctly when five or more bits are received incorrectly. The decision whether a bit is received successfully or not is made by calculating the probability of bit error. The bit error rate is instantly recalculated for all packets being transmitted whenever a new package is arriving simultaneously.

The capture effect is also implemented, but it is assumed that at least one of the packets is lost when two packets with the same SF are transmitted simultaneously. That occurs regardless of the time interval in which the packets coincide on the channel. This is a disadvantage of Module II over Module I, since its implemented physical layer becomes more rigid in this aspect. The MAC layer is implemented in classes *LoRaNetDevice*, *LoRagwNetDevice*, *MacHeader*, *LoRaMacCommand*, and *LoRaRadioEnergyMode*. More information about how these classes work can be found in the original article [10].

One of the advantages of Module II is its IP network implementation for communication between GW and NS, whereas in the other modules, GW and NS are directly linked, or the NS is not present at all. It can also simulate interference from other protocols, a feature that Module I lacks. Other features of the module include the support for ACK commands and slot time reservation for Downlink (DL) messages, a behavior of Class B devices. It also supports MAC commands and has the functionality to increment the SF of an ED after unsuccessful ACK messages. This is an example of an ADR algorithm that has been proposed in the literature. Finally, it supports the *energy* framework from ns-3.

The authors have published three works using their module. In [10], where they also introduced the module, they proposed different scenarios to evaluate the performance of LoRaWAN. They simulate 100, 500, and 1000 EDs distributed in a circle around a central gateway, without ACK messages. Then, they repeat the simulation with seven GWs, spread in a hexagonal grid, and then with a single GW again, but with ACK messages. By evaluating the Packet Error Ratio they concluded that LoRaWAN networks poorly scale when ACK messages are used, and that the use of multiple GWs greatly improves network performance.

In [47], the authors present RS-LoRa, a novel MAC protocol to improve reliability and scalability of LoRaWAN. By using a lightweight scheduling scheme, the GWs guide the EDs in their range to use different SF, enabling simultaneous transmission and reducing packet error. In [48], they use the module to propose a scheme for power and SF allocation in long-range networks.

C. Module III

Module III was developed by researchers from University of Ghent, Belgium, and is available at [49].

For the PHY implementation, the authors developed an error model based on a series of Matlab simulations that measured bit error rate for various LoRa PHY configurations over an Additive White Gaussian Noise (AWGN) channel. Then, they used the ns-3 class *SpectrumPhy* to build the PHY layer, thus allowing for inter-protocol interference. They modeled the execution flow of the devices as a finite state machine and chose to not differentiate the physical layers of ED and GW. This means differences in their transceiver design are not taken into consideration. The MAC layer, however, is different for ED and GW, as they operate in a very different way. The MAC class handles packet queuing, receive windows, acknowledge messages and re-transmissions. They also implemented a simplistic model of the NS, which, like in Module I, is directly connected to the GW and does not receive packets through an IP network.

Further details on how the module was conceived and how it works can be found in [11]. In this work, the authors used the module to evaluate the scalability of LoRa networks. First, they studied how to best assign the SF to EDs, arriving in a solution based on Packer Error Ratio (PER) thresholds. They used a scenario with a single central gateway and a varying number of EDs.

Then, they evaluate the impact of confirmed messages in the Uplink (UL) and DL. Results show that while the use of ACK messages severely hinders the packet delivery ratio on uplink, the difference in performance is negligible for confirmed data in downlink.

In [50] the module is used to study a new algorithm for assigning SFs. Results show that the probability of uplink data delivery increases by 20% to 40% while the cost of power consumption increases only by 1% to 8%. With the same purpose of improving scalability, [51] shows that a new method of synchronization and window scheduling for class A devices increases the packet delivery ratio by 7% to 30% with no bad influences on power consumption.

Finally, the first implementation and simulation of LoRaWAN class B were made by extending Module III. In [52], the authors show their simulations and present an evaluation of the scalability limits of Class B. The module is also used in [53] to simulate a strategy of data gathering of air pollutants in an urban environment. The authors compare different IoT solutions, such as Bluetooth, Wi-Fi, Zigbee, cellular networks, and LoRaWAN, concluding that the latter

is the best for combining long-range and low costs. Then, they implemented the proposed system in ns-3, showcasing its performance in terms of packet delivery ratio.

D. Module IV

The final and most recent module was created by researchers from the Université Grenoble Alpes, in France. It is available at [54]. In their article [12], the authors state that their module was developed based on an open-source code of LoRa Physical layer and LoRaWAN. They also leveraged existing ns-3 code, using and modifying the classes related to ALOHA access method, the spectrum module and the energy framework.

Then, the authors validated their module by comparing the simulation results with their LoRa testbed data. The outcome was very encouraging, as the simulation results are very accurate to the measurements performed. They also compared the module to the measurements reported in the literature, with similar results.

Finally, the authors used the module to evaluate the improvements in performance when contention based medium access methods, such as Carrier Sense Multiple Access (CSMA), are used instead of LoRaWAN standard method. In CSMA, when a device needs to send data, it uses the Listen Before Talking (LBT) principle to assess whether the channel is already in use at the time. If positive, it backs off, going to sleep mode for a random time window before trying again. If the channel is clear, it can start its transmission. This can reduce packet collisions at reception, improving the packet delivery rate, at the cost of more energy consumption due to the listening operation.

The authors implemented in their simulator the CSMA protocol, along with a variant called CSMA-x. In CSMA-x, the device listens to the channel for x ms before transmitting. If it detects occupancy, it backs off similarly to regular CSMA. The authors evaluated the packet delivery ratio for a network with LoRaWAN, CSMA and CSMA-10. The results show that for denser networks, CSMA and CSMA-10 performs better, since a high number of devices results in many collisions in an ALOHA-like protocol. They also evaluated the impact of these methods on energy consumption, showing that the contention based strategies actually achieve a better energy performance as the networks become denser.

Despite this, the module available in [12] lacks their implementation of CSMA and CSMA-x, as well as any sample code on how to interface with the energy framework. Furthermore, no additional documentation is provided, and the code is not didactically commented. These are some of the reasons for our problems in using Module IV.

IV. Module Installation and Testing

In this Section, we describe our experiences when testing the four modules. Simulations were performed in almost all modules for scenarios with a fixed SF 7, fixed SF 12, and SF distribution methods already available in each module. Curves of Packet Delivery Ratio (PDR) in function of the distance from the GW were obtained for each scenario in every module, except for module IV, as we could not repeat the simulations due to issues with the available code. Table VI shows the parameters of the general scenario, remained the same in each module. EDs are uniformly distributed randomly around a single gateway. The average time between packets was defined as 10 minutes in the first three modules. The EDs sends its packets at a random time and each module uses a different distribution model. Module I uses a model based on specification TR 45820 [56]. Modules II and III follow a Poisson-like model. The only scenario characteristic that changes from one module to another is the propagation model. In each module, we choose the most complex propagation model (in terms of shadowing, penetration loss and fading model) available in their code. Table VII shows the differences.

A. Module I

Installation of Module I is accomplished by cloning the repository [34] into the *src* directory of an ordinary ns-3 installation, and performing `configure` and `build` commands. The module provides a text file that briefly describes the PHY and MAC layer models. It also explains some of the module's features and limitations, and provides a list of available trace sources. This is very useful for users who need to keep track of different events through the simulation. A class diagram of the module is provided, as well. It aids in understanding how the code is structured. Additionally, the authors created a chat in Gitter [57], where anyone can ask a question about the module's usage.

The code itself is well organized, and most of the classes have some comment explaining their function. The module closely follows ns-3 coding style [58], using "helpers" objects to setup and configure the network. This makes Module I the most user-friendly of all implementations.

Module I comes with six example files showcasing how to setup a simulation. The first, *simple-lorawan-network-example.cc*, illustrates what steps should be done to send one packet from an ED to a gateway. This example follows the usual ns-3 steps for a simulation of a wireless network. First, the channel models are created using "helpers" classes. Then, a network topology is formed by creating "nodes". A PHY and MAC behavior is assigned to these nodes, as well as an application that allows generating data. Finally, the simulation is started.

While very simple, since a network composed of a single ED and GW is created, this example is useful to understand what classes are involved in the process of generating and sending a packet. Logging is enabled by default, which means that the user can see each of the methods and classes being called by the program during execution. Overall, the example can be used to acquire a better understanding of the module's inner workings. This network is used on the *energy-model-example* example to show how the Lora energy model works, in which the only output is the device's remaining energy.

The second example, *complete-lorawan-network-example.cc*, showcases how to configure a more complex network, with thousands of EDs and multiple GWs. EDs and GWs are placed randomly in a circle of a defined radius. Each ED is

TABLE V
NS-3 MODULES FOR LORA.

Module	URL	Used in	*energy* Framework	ACK Support	Multiple GW	Network Server	Documentation	Other Features
I	[34]	[9], [30], [38], [40], [42]–[45], [55]	Yes	Yes	Yes//Under development	Simple	Excellent	Great support by developers, correlated shadowing model
I*	[31]	[9], [30], [38], [40], [42]–[45], [55]	Yes	No	No	Simple	Excellent	Energy framework based on real measurements
II	[46]	[10], [30], [47]	Yes	Yes	Yes	Through IP	Good	Provides base class to implement applications, highly configurable NS
II*	[32]	[10], [30], [47]	Yes	Yes	Yes	Through IP	Good	Energy framework based on real measurements
III	[49]	[11], [30], [50]–[53]	No	Yes	Yes	Simple	Good	Provides many tracing examples
III*	[33]	[11], [30], [50]–[53]	Yes	Yes	Yes	Simple	Good	Energy framework based on real measurements
IV	[54]	[12]	Yes	No	No	None	Poor	Module validated with measurements

TABLE VI
SYSTEM-LEVEL PARAMETERS FOR SIMULATIONS WITH ALL MODULES.

Parameter	Value
Cell radius (meters)	100 - 19000
Number of End-Devices	100 - 500
Number of Gateways	1
Packet Size (bytes)	51
Time between Packets (minutes)	10
Simulation Time (hours)	10
Transmit power (p_m)	14dBm
Frequency carrier	868.1MHz
Bandwidth (B)	125kHz

TABLE VII
PROPAGATION MODEL USED WITH THE MODULES

Parameter	Module I
Propagation Loss Model	$L = 120.5 + 10 * 3.76 * log_{10}R$
Building Penetration Loss	Yes
Shadowing	Yes
Parameter	Module II
Propagation Loss Model	Okumura Hata + Nakagami
Building Penetration Loss	No
Shadowing	No
Parameter	Module III
Propagation Loss Model	$L = 46.6777 + 10 * 3 * log_{10}R$
Building Penetration Loss	No
Shadowing	No

assigned an SF based on the received power level computed by the *link measurement model* described in Section III-A. A channel model considering only path loss is instantiated, although the option of building propagation loss is already implemented, but it is disabled by default. The EDs are setup with an application that periodically sends messages, and the simulation is started. The metrics examined are the packets sent and packets received by at least one gateway. Although the module can create a network with multiple GWs, some metrics will not be easily obtained correctly, since the packet tracking system public available is not meant to be used with multiple gateways. The authors claim that for some cases like data about MAC layer performance, the module will work properly; in other cases, the user can write its own packet tracking system [57]. Despite these issues, the authors affirm they are working on a packet tracking system available for multiple gateways [57]. Meanwhile, the example is useful since it presents how to build a more complex network and how to obtain certain metrics. The *aloha-throughput.cc* example creates a similar network, but some configurations are changed to analyze the network performance if the system uses a ALOHA protocol, when collisions imply the loss of both packets.

Another example is *network-server-example.cc*. It creates a network composed of one ED, GW, and NS that introduces how to setup these components to interact with each other. Similarly to the first example, it showcases the step-by-step process that the simulation takes to generate and transmit a packet from the ED through the GW, and how the NS interprets it.

The *adr-example.cc* creates a simple network with fixed and mobile EDs. Through a server-side application, an ADR algorithm sets up the Spreading Factors of EDs. The examined output are the packets sent and packets received by at least one GW. Some files are also generated, which display essential metrics obtained periodically during the simulation: general MAC performance (*globalPerformance.txt*), PHY performance per GW (*phyPerformance.txt*) and position, data rate and TX power of every ED (*nodeData.txt*). Through these files, we can see changes in the network over time due to the ADR algorithm.

We modified the parameters of the second example to obtain the curves presented in Fig.2 and Fig.3. In each module, we

choose one example to parameterize and generate the curves with SF 7, SF 12 and the available ADR mechanism. Other parameters for each module are presented in Table VI.

In Fig.2, it can be seen that for SF 7, the packet delivery ratio (PDR) starts slightly better than SF 12, but PDR declines much faster with distance from the GW for SF 7. The best performance is reached by the ADR mechanism available in Module I, where the SFs are assigned to the EDs by the network server based on the SNR of past packets. For 500 EDs, the difference between SF 7 and SF 12 is much more evident. With SF 7, PDR starts high but declines quickly as the distance increases, very similar to the 100 EDs scenario. With SF 12, PDR starts much smaller, but its value continues similar when ranging the distance, different from the 100 EDs scenario. Since SF 12 provides a long range, a lot of interference will occur when many EDs are spread in a small area. As distance from the GW increases, less interference happens, but more EDs will be out of range. Since SF 12 has a long range, these two effects combined a more stable tendency, as shown in Fig.3. Finally, for the ADR mechanism, PDR is almost the same with 100 and 500 EDs.

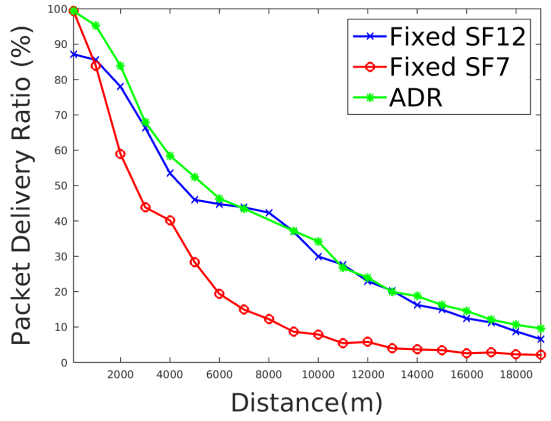

Fig. 2. Module I - Packet Delivery Ratio for different SF assignment schemes with 100 EDs.

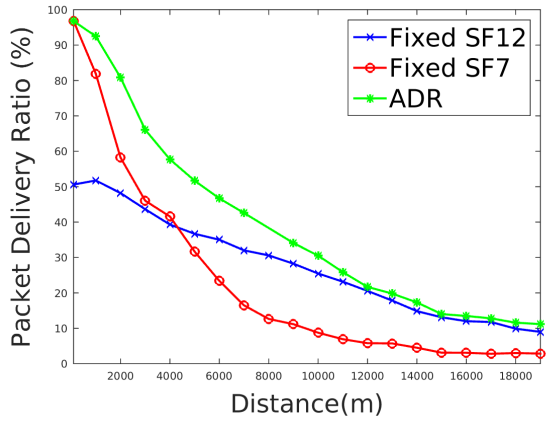

Fig. 3. Module I - Packet Delivery Ratio for different SF assignment schemes with 500 EDs.

B. Module II

Module II installation is accomplished in the same way as the previous module, except for some line codes that have to be added for the correct building of the module. The most recent version of the module requires compiler support for the ISO C++2014 standard, while ns-3 uses ISO C++2011 as default. That can be changed by modifying the *wscript* of ns-3. The repository is found in [46].

Unlike Module I, there is no documentation explaining the main features of the module. The article that introduces it, which presents a class diagram of the module, serves as the best reference to understand the implementation. Besides that, the code is well organized, most of the classes have some commentary explaining their functionality, and the code is compliant with ns-3 standards.

Three examples are provided. The first one, *lora_battery.cc*, shows the step-by-step process to configure a network with EDs and a GW, setup channel models, set mobility and energy models on the nodes, and how to use the tracing system to obtain some metrics. The simulation shows the battery depletion of the devices as they transmit and also some data related to packet transmissions and the number of re-transmissions. Overall, it illustrates satisfactorily how to use the module. A similar code is *rs_example.cc*, but the proposal is to show data related to packet transmissions for a different type of LoRa network, which allows beaconed scheduling. The example *lora.cc* is a more general example, which allows us to set some network configurations, like choosing one of the server's application options, if we use ACK messages, LoRa or rs-LoRa MAC layer, add interference and monitor power consumption. It illustrates properly how to use different module functions.

We modified the parameters of the first example to generate the curves of Fig.4 and Fig.5, which happen to be very different from Module I. With 100 EDs and fixed SF 7, the PDR already starts considerably low (at 60%) and declines quickly as distance from GW increases. The interference is higher, even for SF 7, since this module models the capture effect differently. Since SF 7 has a low range, the curve declines quickly as the distance ED-GW increases. For a fixed SF 12, interference is occurring for small distances as well, but more devices are in the coverage range, and it remains almost constant as the distance increases. In contrast with the Module I, for 100 EDs the SF 12 has a better result then SF 7. With a different propagation model, the better coverage of SF 12 has more influence than the smaller interference of SF 7, even for shorter distances. With 500 devices, as shown in Fig.5, too many devices in small distances produce more interference with SF12 then SF 7 due to its long range, leading to the low PDR values for both cases.

Module II provides an ADR mechanism, were all devices start with SF 7, and then EDs SF are adjusted by the network server based on the channel conditions. If a device sends 91 packets and does not receive a message from GW, the ED automatically increases its SF. We call this ADR mechanism "Fixed SF7 + ADR". In Fig.4, we can see that the ADR presents the best PDR for up to 6000 m, and it declines with

distance, but still performing better than the fixed SF 7 setting at longer distances. Regarding the scenario with 500 EDs, presented in Fig.5, the ADR mechanism performs better than both Fixed SF 12 and SF 7 settings, showing its efficiency regarding the PDR. The ADR proposal of Module II performs better than the standard ADR mechanism implemented by Module II e III.

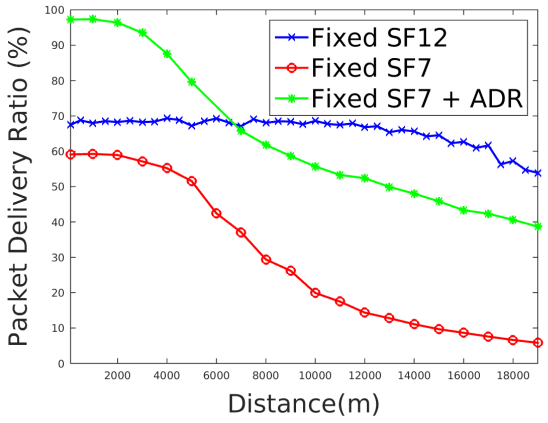

Fig. 4. Module II - Packet Delivery Ratio for different SF assignment schemes with 100 EDs.

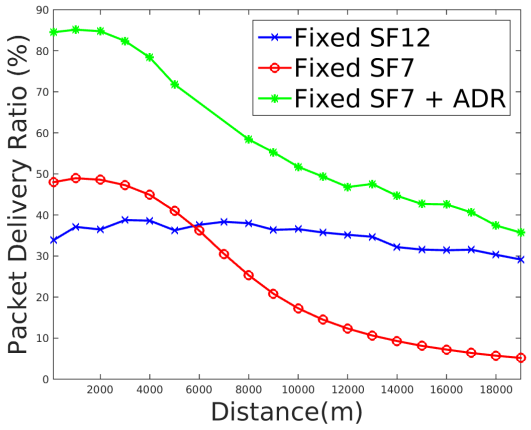

Fig. 5. Module II - Packet Delivery Ratio for different SF assignment schemes with 500 EDs.

C. Module III

Module III is available to download in [49]. Unlike the other modules, this repository contains the entire ns-3 directory with the module inside. This makes it more difficult to port the Module III to different ns-3 installations, although we verified that such porting is possible.

Module III includes a short text file with a brief explanation about it. The module comes with many example codes. However, most of those are broken and will cause ns-3 to fail when trying to build the simulator. The only example that compiles is *lorawan-tracing-example.cc*. The others must be removed from the example folder, or examples must be disabled in ns-3 configuration so that it can successfully be built.

The authors state in their article [11] that this example was used for all simulations in their work. The code is indeed pervasive and seems to showcase almost all of the features of the module. This makes it very hard to understand the code, especially considering its minimal documentation.

Nevertheless, the example can be used as a reference to elaborate one's own simulation since it showcases how to setup the network and the tracing sources. Upon execution, the examples generate several files containing data from the simulation. It is worth mentioning that this example uses some functions available only in C++11. The default ns-3 code avoids using these functions, so many ns-3 installations use compilers incompatible with C++11. Version 5.0 or higher of *gcc* is required to run the example.

We changed the example to generate the curves in Fig.6 and Fig.7. The graphics are similar to those of Module I, however interference has more impact with SF 12. For SF 7, the PDR tendency starts in a equal point to the ADR, but it quickly decays in function of the distance from the GW due to its low range in both scenarios. For fixed SF 12 and 100 EDs, as shown in Fig.6, the PDR starts at just around 70% due to interference between devices closer to each other and with long range. Also, up to approximately 7000 m the PDR remains almost constant, then it starts to decay as distance increases, remaining higher than SF 7 case. For 500 EDs, shown in Fig.7, interference is even higher for the fixed SF 12 scenario, and the PDR starts around 30%, decreasing to 20% around 6000 m. We can also observe the tendency of the PDR for a random SF and the ADR mechanism implemented in Module III, where it presents the best performance in both 100- and 500-ED scenarios.

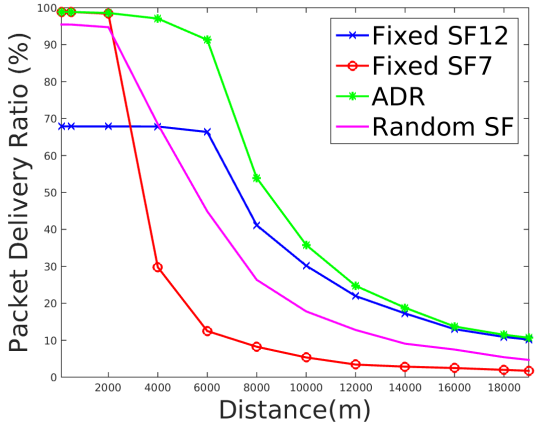

Fig. 6. Module III - Packet Delivery Ratio for different SF assignment schemes with 100 EDs.

D. Module IV

Module IV is found in [54]. It must be cloned to the *src* directory of the ns-3 installation, and its folder needs to be renamed to *lora* before ns-3 `configure` and `build` commands are executed. The reason for renaming the folder is

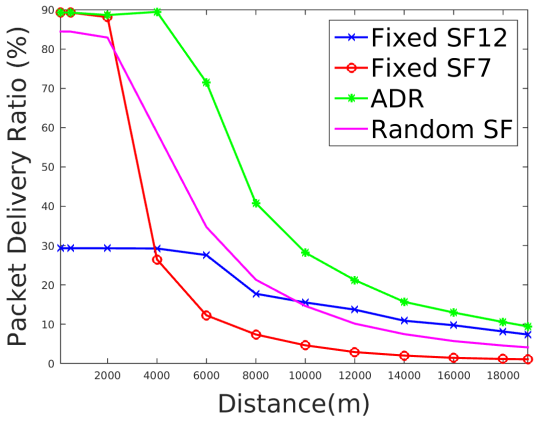

Fig. 7. Module III - Packet Delivery Ratio for different SF assignment schemes with 500 EDs.

because the repository's name does not match the name given to the module in the building scripts.

This module comes with a single example. This example, by default, simulates a network with ten EDs and one GW, with a simulation duration of 100 s. The EDs send messages, and the number of messages that successfully reach the GW is shown. The code defines a *Loraexample* to hold all methods to setup and start the simulation, also some variables like the number of nodes and simulation time. We found that no matter the value assigned to the *size* member of the *Loraexample* class, which represents the number of EDs, the number of packets that are shown to be received is 10.

There is no user-friendly documentation about the code and its configuration, hindering an easy simulation campaign setup. A series of methods are defined and called on the example instead of using helpers to configure the network. Finally, the classes related to CSMA and the energy framework are missing. Due to the issues listed above, we could not produce results for this module.

V. Conclusion

In this paper, we presented an overview of the available tools for simulation of LoRaWAN in ns-3. The development of those simulators is essential to aid the community to better understand and assess this LPWAN protocol.

In our survey, we found Module I to be the easiest to work with, as an excellent documentation and organized code are provided. Although many features are not available yet, such as IP Network, complete NS implementation, and interference with other protocols, the authors have shown they are still working on the module, and provide excellent support to other users. It is also the preferred module by the community, as it is the most used outside of the authors' works.

Module II provides the most LoRaWAN features, with special mention to the more realistic implementation of the NS and for addressing some shortcomings of the other modules. It also allows for the implementation of algorithms on the server side. However, its documentation is found only in research papers.

Module III also lacks some features, since the NS implementation is simple. Its original paper [11] provides a detailed description of the module, although the code in the repository requires some work to compile correctly.

Finally, Module IV was the only one validate through real measurements. However, missing functions in the repository and no additional documentation, as well as a broken sample code, are factors that restrict the simulation campaigns.

Overall, we believe Module I is the most promising, considering the continuous support from the authors and popularity in other researches. However, if one wishes to study inter-protocol interference, Module II and III are the better choices. If multiple gateways are necessary, the use of Module II and III could be easier, depending on which metrics are required, although extending Module I is probably the best approach. In addition, Module II is the better choice for evaluating server-side algorithms, as its NS has the most complete implementation of all modules.

Finally, we provide a list of open issues that are important to the development of more realistic simulations. Module I does not cover inter-protocol interference and needs a better implementation of the Network Server to better simulate advanced features like different Adaptive Data Rate (ADR) algorithms, to support device joining procedures and to respond to the ED's MAC commands. Module II is the most complete of all modules, but still lacks some MAC commands, since only the *LinkAdrReq* and *LinkAdrAns* commands are implemented. Module III still need integration with an energy framework and also needs a more realistic implementation of the Network Server. Unavailable features in all modules includes integration with more realistic propagation models, implementation of classes B and C devices, and the modeling of interference between partially overlapping channels. Since the modules presented are open-source, we believe those issues can be explored by the academic community to greatly improve the available tools for LoRaWAN simulation.

References

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communications Surveys and Tutorials*, 2015, doi: 10.1109/ITST.2015.7377400.

[2] ITU-R, *Report ITU-R M.2410-0 Minimum requirements related to technical performance for IMT - 2020 radio interface(s)*, International Telecommunication Union Std., 2017.

[3] F. Montori, L. Bedogni, M. Di Felice, and L. Bononi, "Machine-to-machine wireless communication technologies for the Internet of Things: Taxonomy, comparison and open issues," 2018, doi: 10.1016/j.pmcj.2018.08.002.

[4] K. Staniec and M. Kowal, "LoRa Performance under Variable Interference and Heavy-Multipath Conditions," *Wireless Communications and Mobile Computing*, vol. 2018, pp. 1–9, 04 2018, doi: 10.1155/2018/6931083.

[5] J. Petäjäjärvi, K. Mikhaylov, A. Roivainen, T. Hänninen, and M. Pettissalo, "On the coverage of LPWANs: Range evaluation and channel attenuation model for LoRa technology," in *2015 14th International Conference on ITS Telecommunications, ITST 2015*, 2016, doi: 10.1109/ITST.2015.7377400.

[6] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, "Understanding the Limits of LoRaWAN," *IEEE Communications Magazine*, 2017, doi: 10.1109/MCOM.2017.1600613.

[7] M. Bor, U. Roedig, T. Voigt, and J. M. Alonso, "Do LoRa low-power wide-area networks scale?" in *MSWiM 2016 - Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2016, doi: 10.1145/2988287.2989163.

[8] M. Kabir, S. Islam, M. Hossain, and S. Hossain, "Detail comparison of network simulators," *International Journal of Scientific & Engineering Research*, 2014.

[9] D. Magrin, M. Centenaro, and L. Vangelista, "Performance evaluation of LoRa networks in a smart city scenario," in *IEEE International Conference on Communications*, 2017, doi: 10.1109/ICC.2017.7996384.

[10] B. Reynders, Q. Wang, and S. Pollin, "A LoRaWAN module for ns-3: Implementation and evaluation," in *ACM International Conference Proceeding Series*, 2018, doi: 10.1145/3199902.3199913.

[11] F. Van Den Abeele, J. Haxhibeqiri, I. Moerman, and J. Hoebeke, "Scalability Analysis of Large-Scale LoRaWAN Networks in ns-3," *IEEE Internet of Things Journal*, 2017, doi: 10.1109/JIOT.2017.2768498.

[12] T. H. To and A. Duda, "Simulation of LoRa in NS-3: Improving LoRa Performance with CSMA," in *IEEE International Conference on Communications*, 2018, doi: 10.1109/ICC.2018.8422800.

[13] M. Slabicki, G. Premsankar, and M. Di Francesco, "Adaptive configuration of lora networks for dense IoT deployments," in *IEEE/IFIP Network Operations and Management Symposium: Cognitive Management in a Cyber World, NOMS 2018*, 2018, doi: 10.1109/NOMS.2018.8406255.

[14] M. Bor, U. Roedig, T. Voigt, and J. M. Alonso, "Do LoRa low-power wide-area networks scale?" in *MSWiM 2016 - Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2016, doi: 10.1145/2988287.2989163.

[15] ns-3. (2016) ns-3 website (http://code.nsnam.org). [Online]. Available: http://code.nsnam.org/

[16] J. Haxhibeqiri, E. De Poorter, I. Moerman, and J. Hoebeke, "A survey of LoRaWAN for IoT: From technology to application," 2018, doi: 10.3390/s18113995.

[17] M. Saari, A. Muzaffar Bin Baharudin, P. Sillberg, S. Hyrynsalmi, and W. Yan, "LoRa - A survey of recent research trends," in *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2018 - Proceedings*, 2018, doi: 10.23919/MIPRO.2018.8400161.

[18] A. Augustin, J. Yi, T. Clausen, and W. M. Townsley, "A study of Lora: Long range & low power networks for the internet of things," *Sensors (Switzerland)*, 2016, doi: 10.3390/s16091466.

[19] J. De Carvalho Silva, J. J. Rodrigues, A. M. Alberti, P. Solic, and A. L. Aquino, "LoRaWAN - A low power WAN protocol for Internet of Things: A review and opportunities," in *2017 2nd International Multidisciplinary Conference on Computer and Energy Science, SpliTech 2017*, 2017.

[20] A. Lavric and V. Popa, "Internet of Things and LoRa™ Low-Power Wide-Area Networks: A survey," in *ISSCS 2017 - International Symposium on Signals, Circuits and Systems*, 2017, doi: 10.1109/ISSCS.2017.8034915.

[21] LoRa Alliance Technical Committee, *LoRaWAN™1.1 Specification*, LoRa Alliance Std., 2017.

[22] 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*, vol. 1, p. 150597, 10 2015, doi: 10.4108/eai.26-10-2015.150597.

[23] S. Corporation. (2015) AN1200.22 LoRa™ Modulation Basics. [Online]. Available: www.semtech.com/uploads/documents/an1200.22.pdf

[24] M. Knight. (2016) Decoding LoRa: Exploring Next-Generation Wireless. [Online]. Available: https://github.com/matt-knight/research

[25] L. Vangelista, "Frequency Shift Chirp Modulation: The LoRa Modulation," *IEEE Signal Processing Letters*, 2017, doi: 10.1109/LSP.2017.2762960.

[26] D. Magrin. (2016) Network level performances of a LoRa system. [Online]. Available: http://tesi.cab.unipd.it/53740/1/dissertation.pdf

[27] LoRa Alliance Technical Committee Regional Parameters Workgroup, *LoRaWAN 1.1 Regional Parameters*, LoRa Alliance Std., 2018.

[28] S. Corporation, "Lorawan-simple rate adaptation recommended algorithm," https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwi515OEsYXsAhWxBtQKHW2OBTEQFjADegQIAxAB&url=https\%3A\%2F\%2Fwww.thethingsnetwork.org\%2Fforum\%2Fuploads\%2Fdefault\%2Foriginal\%2F2X\%2F7\%2F7480e044aa93a54a910dab8ef0adfb5f515d14a1.pdf&usg=AOvVaw3ke_sNL8JRR1Rj49-S1kKt, acessed on em 25/09/2020.

[29] J. Finnegan, S. Brown, and R. Farrell, "Evaluating the Scalability of LoRa WanGateways for Class B Communication in ns-3," in *2018 IEEE Conference on Standards for Communications and Networking, CSCN 2018*, 2018, doi: 10.1109/CSCN.2018.8581759.

[30] ——, "Modeling the energy consumption of lorawan in ns-3 based on real world measurements," in *2018 Global Information Infrastructure and Networking Symposium, GIIS 2018*, 2019, doi: 10.1109/GIIS.2018.8635786.

[31] (2018) ns3 lora module. [Online]. Available: https://github.com/ConstantJoe/signtlabdei-lorawan-with-energy-model

[32] (2018) ns3 lora module. [Online]. Available: https://github.com/ConstantJoe/ku-leuven-lorawan-with-energy-model

[33] (2018) ns3 lora module. [Online]. Available: https://github.com/ConstantJoe/imec-idlab-lorawan-with-energy-model

[34] D. Magrin, M. Capuzzo, S. Romagnolo, and M. Luvisotto. (2017) LoRaWAN ns-3 module. [Online]. Available: https://github.com/signetlabdei/lorawan

[35] Semtech. (2017) Sx1301 datasheet. [Online]. Available: https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/44000000MDnR/Et1KWLCuNDI6MDagfSPAvqqp.Y869Flgs1LleWyfjDY

[36] M. Bor, J. Vidler, and U. Roedig, "LoRa for the Internet of Things," in *Proceedings of the 2016 International Conference on Embedded Wireless Systems and Networks*, 2016.

[37] D. Magrin, M. Capuzzo, S. Romagnolo, and M. Luvisotto. (2017) LoRaWAN ns-3 module. [Online]. Available: https://github.com/DvdMgr/lorawan

[38] 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*, 2018, doi: 10.23919/MedHocNet.2018.8407095.

[39] D. Magrin, M. Centenaro, and L. Vangelista, "Performance evaluation of LoRa networks in a smart city scenario," in *IEEE International Conference on Communications*, 2017, doi: 10.1109/ICC.2017.7996384.

[40] Y. Hasegawa and K. Suzuki, "A Multi-User ACK-Aggregation Method for Large-Scale Reliable LoRaWAN Service," in *IEEE International Conference on Communications*, 2019, doi: 10.1109/ICC.2019.8761253.

[41] M. Capuzzo, D. Magrin, and A. Zanella, "Mathematical Modeling of LoRa WAN Performance with Bi-directional Traffic," in *2018 IEEE Global Communications Conference, GLOBECOM 2018 - Proceedings*, 2018, doi: 10.1109/GLOCOM.2018.8647351.

[42] Y. Oukessou, M. Baslam, and M. Oukessou, "LPWANIEEE 802.11ah and LoRaWAN capacity simulation analysis comparison using NS-3," in *Proceedings of the 2018 International Conference on Optimization and Applications, ICOA 2018*, 2018, doi: 10.1109/ICOA.2018.8370592.

[43] N. Kouvelas, V. Rao, and R. R. Prasad, "Employing p-CSMA on a LoRa Network Simulator," 2018.

[44] M. Luvisotto, F. Tramarin, L. Vangelista, and S. Vitturi, "On the Use of LoRaWAN for Indoor Industrial IoT Applications," *Wireless Communications and Mobile Computing*, 2018, doi: 10.1155/2018/3982646.

[45] S. Dawaliby, A. Bradai, and Y. Pousset, "Adaptive dynamic network slicing in LoRa networks," *Future Generation Computer Systems*, 2019, doi: 10.1016/j.future.2019.01.042.

[46] (2017) ns3 lora module. [Online]. Available: https://github.com/networkedsystems/lora-ns3

[47] B. Reynders, Q. Wang, P. Tuset-Peiro, X. Vilajosana, and S. Pollin, "Improving reliability and scalability of LoRaWANs through lightweight scheduling," *IEEE Internet of Things Journal*, 2018, doi: 10.1109/JIOT.2018.2815150.

[48] B. Reynders, W. Meert, and S. Pollin, "Power and spreading factor control in low power wide area networks," in *IEEE International Conference on Communications*, 2017, doi: 10.1109/ICC.2017.7996380.

[49] Floris Van den Abeele, Jetmir Haxhibeqiri, Ingrid Moerman, and Jeroen Hoebeke. (2017) Lorawan ns-3 module. [Online]. Available: https://github.com/imec-idlab/ns-3-dev-git/tree/lorawan

[50] A. Tiurlikova, N. Stepanov, and K. Mikhaylov, "Method of Assigning Spreading Factor to Improve the Scalability of the LoRaWan Wide Area Network," in *International Congress on Ultra Modern Telecommunications and Control Systems and Workshops*, 2019, doi: 10.1109/ICUMT.2018.8631273.

[51] J. Haxhibeqiri, I. Moerman, and J. Hoebeke, "Low overhead scheduling of LoRa transmissions for improved scalability," *IEEE Internet of Things Journal*, 2019, doi: 10.1109/JIOT.2018.2878942.

[52] J. Finnegan, S. Brown, and R. Farrell, "Evaluating the Scalability of LoRa WanGateways for Class B Communication in ns-3," in *2018 IEEE Conference on Standards for Communications and Networking, CSCN 2018*, 2018, doi: 10.1109/CSCN.2018.8581759.

[53] H. Rahim, C. Ghazel, and L. A. Saidane, "An Alternative Data Gathering of the Air Pollutants in the Urban Environment using LoRa and LoRaWAN," in *2018 14th International Wireless Communications and Mobile Computing Conference, IWCMC 2018*, 2018, doi: 10.1109/IWCMC.2018.8450329.
[54] D. M. M. C. R. M. Luvisotto. (2017) LoRaWAN ns-3 module. [Online]. Available: https://github.com/drakkar-lig/lora-ns3-module
[55] A. D. Prajanti, B. Wahyuaji, F. B. Rukmana, R. Harwahyu, and R. F. Sari, "Performance Analysis of LoRa WANTechnology for Optimum Deployment of Jakarta Smart City," in *2018 2nd International Conference on Informatics and Computational Sciences, ICICoS 2018*, 2018, doi: 10.1109/ICICOS.2018.8621803.
[56] iTecTec. 3gpp tr 45.820 – cellular system support for ultra-low complexity and low throughput internet of things (ciot). [Online]. Available: https://itectec.com/archive/3gpp-specification-tr-45-820
[57] Module I gitter char. [Online]. Available: https://gitter.im/ns-3-lorawan
[58] ns-3 Coding Style Guidelines. [Online]. Available: https://www.nsnam.org/develop/contributing-code

Jéssika C. da Silva received her degree in Eletrical Engineering from the Federal University of Rio Grande do Norte (UFRN), Brazil, in 2018. Since 2016 she has participated in research on wireless networks, network simulation and signal processing. She is currently pursuing her M.Sc Electrical and Computer Engineering degree at UFRN, researching improvements to ADR algorithms in LoRaWAN networks.

Daniel de L. Flor received his degree in Electrical Engineering from the Federal University of Rio Grande do Norte (UFRN), Brazil, in 2018. Since 2017 he has participated in research on wireless networks, network simulation and signal processing. He is currently pursuing his M.Sc Electrical and Computer Engineering degree at UFRN, researching the characteristics of acoustic noise in intra-vehicle environments.

Vicente A. de Sousa Jr. received his B.S., M.S and Ph.D. degrees in Electrical Engineer from Federal University of Ceará (UFC), Fortaleza, CE, Brazil, in 2001, 2002 and 2009, respectively. Between 2001 and 2006, he developed solutions to UMTS/WLAN interworking for UFC and Ericsson of Brazil. Between 2006 and 2010, he contributed to WIMAX standardization and Nokia's product as a researcher at Institute of Technological Development (INDT). Dr. Sousa is now a professor at Federal University of Rio Grande do Norte (UFRN), Brazil.

Níbia Souza Bezerra received the B.Sc. degree in Computer Engineering from Amazonas State University in 2008 and her M.Sc. in Teleinformatics Engineering from the Federal University of Ceará in 2011. From 2008 to 2011 she worked for Nokia Technology Institute (INdT) as a researcher, doing research related to wireless communications. From 2011 to 2013 she was a researcher at the Wireless Telecom Research Group (GTEL), Fortaleza, Brazil, where she worked in projects in cooperation with Ericsson Research. She is currently pursuing her Ph.D. in Computer Sciences at Luleå University of Technology in Skellefteå, Sweden. She currently works as Senior Software Developer at TietoEvry AB in Skellefteå. Her research interests include mobility for machine-type communication, smart cities, the Internet of Things (IoT) for current wireless networks and future 5G and 6G.

Alvaro A. M. de Medeiros received the degree in electrical engineering from the Federal University of Rio Grande do Norte, Brazil, in 2000, and the M.Sc. and Ph.D. degrees in electrical engineering from the University of Campinas, Brazil, in 2002 and 2007, respectively. From 2007 to 2010, he was a Research Specialist at Nokia Institute of Technology, Brazil, and Research and Development Center, Brazil. He is currently an Associate Professor at Federal University of Juiz de Fora, Brazil.