A Survey of LoRaWAN Simulation Tools in ns-3

Daniel de L. Flor, Jéssika C. da Silva Vicente A. de Sousa Jr.

Abstract—The Internet of Things (IoT) paradigm gains more importance each day, as the number and variety of connected devices grows. To answer the specific needs of the IoT, many wireless networks standards have been developed. An example is the LoRaWAN, which has gained popularity thanks to the affordable costs of its devices and gateways, as well as wide range. 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. 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 simulators to highlight what scenarios each is more suited for.

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

I. INTRODUCTION

The IoT (Internet of Things), a paradigm in which an enormous number of devices are 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. It finds applications in a wide range of scenarios, from smart buildings and 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 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.

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 km², 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 shown 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 PHY and 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-WPAN) have a short range and low data rates, reducing power consumption of devices. LR-WPAN are focused on applications that connect a relatively small amount of devices in a small area;
- Cellular based networks leverage the infrastructure of the already well-established mobile systems, such as GSM and LTE. They are adapted versions of the traditional cellular networks, in order to support massive connection and low power requirements;
- Low Power Wide Area Networks (LPWAN) are characterized by a coverage in a wide area and by operation in the unlicensed bands. Since they were conceived with IoT applications in mind, many of the design choices in LPWAN take into account the battery restraints.

| Type     | Key Aspects                          | Examples               |
|----------|--------------------------------------|------------------------|
| LR-WPAN  | Low data rates and small number of devices, established technology | Bluetooth, Zigbee, 6LoWPAN |
| Cellular | Uses mobile network infrastructure, high cost to implement | LTE-M, NB-IoT, EC-GSM-IoT |
| LPWAN    | Long range, low power consumption, unlicensed bands operation | 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, together with the low signaling overhead of the protocol, allows for a wide coverage with little energy consumption [3].
LoRaWAN, a LPWAN, 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, allowing 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 of prototyping tools of general and state-of-the-art IoT features, we adopt LoRaWAN as our target enabling technology.

B. Network Simulation Tools

Networks simulators are very important tools for network researching and development. They allow us to perform studies that are difficult or expensive to perform in a real system or to obtain data about these systems in a controlled and reproducible environment. A particular example is the study of LPWAN, which can have thousands of devices connected in a single network. Table II presents five of the most popular simulators for wireless networks. They all have support for different wireless standards (such as the 802.11 family) and models for traffic generation, mobility, channel propagation and energy consumption. More information on these and other simulators can be found in [8].

In this work, we focus on the ns-3 due to its popularity, open source code, continuous support and good documentation. The ns-3 is a discrete-event network simulator primarily used for research and educational purposes [9]. The simulator’s main code is written in C++, with Python being used for bindings. In its core, there is a set of libraries of 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.

Due to these traits and to the popularity of LoRa, 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. In this paper, we present a survey of these simulators, along with their key features and limitations.

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. For instance, it can mean 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 RF chips.

LoRa can also refer to the network of end-devices and gateways that employ LoRa modulation, or to the MAC-layer specification of these networks. Both of these are 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 [10]. It defines physical layer parameters, frame formats, classes of devices, and security and network protocols.

The reminder of this section examines the keys 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 DSSS (Direct Sequence Spread Spectrum) 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 and interference. 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 [11].

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 the 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. In the case of LoRa, the frequency of the chirp varies linearly in the interval $[f_a, f_b]$, which also defines the occupied bandwidth $BW$. If the chirp has a duration $T_s$, the instantaneous frequency $f(t)$ can be defined as

$$f(t) = f_0 + \mu \frac{BW}{T_s} t.$$  

In the expression above, $\mu$ determines whether the signal is a down-chirp ($\mu = -1$) or a up-chirp ($\mu = 1$). If the signal is a up-chirp, the starting frequency $f_0$ will be the lower end ($f_a$) of the interval. Otherwise, the starting frequency is $f_b$.

In the basic CSS modulation, up-chirps and down-chirps are used to represent binary 1s and 0s. LoRa modulation, however, uses a multistate CSS to code up to 12 bits in a single chirp. To achieve this, different symbols are represented by cyclical shifts of the up-chirp signal [12]. These shifts occur at multiples of $\Delta f$, which are known as the “chip” of CSS modulation.

A symbol is represented in the chirp by starting the up-chirp signal at frequency $f_0$: 
TABLE II

| 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 Python| 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

| 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 for 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 6.

At reception, demodulation is performed by multiplying the signal with a 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 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 [14].

In addition to multistate CSS, LoRa employs some encoding schemes before modulation and transmission to further improve its robustness and reliability [15]. 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 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 noise degradation over the bandwidth and the time on air of the packets.

B. LoRaWAN MAC Layer

The architecture of LoRaWAN networks is represented in the Figure 1.

Increasing SF allows for 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 6.

At reception, demodulation is performed by multiplying the signal with a 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 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 [14].

In addition to multistate CSS, LoRa employs some encoding schemes before modulation and transmission to further improve its robustness and reliability [15]. 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 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 noise degradation over the bandwidth and the time on air of the packets.

B. LoRaWAN MAC Layer

The architecture of LoRaWAN networks is represented in the Figure 1.

Fig. 1. Basic topology of LoRaWAN networks. Reproduced from [15].
messages to/from Gateways (GWs), using a LoRa link. The gateways forward end-devices’ 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 costumer’s application locally or remotely. It can also send downlink messages to the EDs via the gateways, if necessary.

As shown in the Figure 1, a 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 messages to the NS. It’s the server’s responsibility to filter duplicate packets and decode them. It also assigns the best gateway to deliver a packet to an ED. Figure 2, which presents the protocol stack of LoRaWAN, shows that the NS and ED are logically linked. The GW is 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 [10], [17]. 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);
- Issue limitations to the EDs 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 behavior 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 regularly 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 ACK (Acknowledge) messages, allowing the NS to confirm a packet’s arrival to a 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 [15], [18] show that the packet loss rate in LoRaWAN networks is lower than that expected of pure ALOHA. This is due to the capture effect, enabled by the quasi-orthogonality of LoRa transmissions.

LoRaWAN also defines several regional parameters that comply with different regulatory standard for the ISM (Industrial, Science and Medical) bands worldwide [19]. 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, provided these are used exclusively within a private network.

### 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. We labeled them as Module I through IV, in order of the date they were made publicly available.

While the Module I was tested in ns-3 version 3.27, the other were tested in ns-3 version 3.28. It is also worth mentioning that all four modules implement only Class A devices, since it is the main behavior of EDs in a IoT scenario.

#### A. Module I

The Module I was developed in 2016 as a part of the Masters Thesis of D. Magrin [15], in The University of Padova, Italy. It is available for download in [20]. The authors developed their simulator by simplifying the transmission chain in two main models. A very detailed description of those models is provided in [15]. 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 SINR at the reception). To accomplish this, the model considers the receivers sensitivity (based on the datasheet of the SX1301 chip, employed in gateways) and the interference introduced by overlapping packet arrivals. The MAC layer implementation provides the structure for MAC commands, and also handles the header formats, the addressing of EDs, the logical channel and the duty cycle management.

Many of these models were implemented by leveraging the already established models in ns-3. For instance, the
Fig. 2. Protocol stack of LoRaWAN. Reproduced from [16].

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.

The main drawback of this implementation is that not all of the features described in the thesis are available in the public repository. Because of this, some of the examples provided do not work properly. This is discussed in the next section. Some of the missing features can be found in older commits and in the development branch of the repository, but not fully integrated.

Other shortcomings include the inability to account for interference of other systems in the ISM band; the lack of integration with the ns-3 energy module, which can be used evaluate power consumption; the lack of downlink transmissions; and the simple implementation of the NS, which lacks advanced features like ADR algorithms and response to EDs MAC commands.

The authors used Module I on two works. In [21], 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. In [22], the authors study the effects of confirmed traffic in LoRa networks, concluding that it can impact performance negatively if not restricted. To do so, the authors implemented ACK messages; this feature, however, is not available in the public repository.

Other researchers have also used the module. In [23], the authors used ns-3 to compare the capacity of LoRaWAN to IEEE 802.11ah, another LPWAN standard, showing that the former performs better on this aspect. In [24], the authors evaluated the improvement in scalability of LoRa networks when p-CSMA, a contention based medium access strategy, is used as the MAC mechanism instead of the ALOHA-like LoRaWAN standard. In [25], the module is used to assess the behavior of LoRaWAN in a typical industrial monitoring scenario. They implemented energy consumption models that interface with the energy framework in ns-3, although they did not make them available.

B. Module II

The Module II was developed by Brecht Teynders and other researchers from KU Leuven University, in Belgium. It is available for download in [26]. The authors stated in [16], their module has been used for two years in LoRaWAN researches.

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 relatively easy to understand the code. The module was submitted to The Workshop on ns-3 (WNS3, 2018) and was accepted for publication [16].

The physical layer is implemented in the Loraphy and Loragrophy, for ED and GW, respectively. The class spectrumsignalparameters is used to setup parameters like the SF and bandwidth. The PHY implementation takes into consideration the interference between transmission with different SF. The MAC layer is implemented in classes LoRaNetDevice, LoRagwNetDevice, MacHeader, LoRaMacCommand, LoRaRadioEnergyMode. More information about how these classes work is found in the original article [16].

One of the advantages of Module II is that it implements the communication between GW and NS over an IP network, whereas in the other modules GW and NS are directly linked, or the NS is not in 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 DL messages, a behavior of Class B devices. It also support MAC commands and includes the functionality to increment the SF of an ED after two unsuccessful ACK messages. This is an example of 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 the module. In [16], where they also introduce the module, they propose 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. They concluded, by evaluating the Packet Error Ratio, that LoRaWAN networks scale poorly when acknowledge messages are used, and that the use of multiple GWs greatly improve network performance. This can be seen in Figure 3, a result from their study.

Fig. 3. Packet Error Rate for EDs distribute in a circle around a single central gateway, with ACK messages on. Reproduced from [16].

In [27], 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 [28], 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 in [29].

For the PHY implementation, the authors developed an error model based on a series of MATLAB simulations that measured bit error for various LoRa PHY configurations over an 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 [30]. 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 Packet Error Ratio (PER) thresholds. The performance of the several SF assignment strategies are resumed in Figure 4. They used a scenario with a single central gateway and a varying number of EDs.

Fig. 4. Packet Error Rate for different SF assignment schemes. Reproduced from [30]. Then, they evaluate the impact of confirmed messages in the upstream and downstream. 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.

The module is also used in [31] 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 in [32]. In their article [18], 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 simulator with their LoRa testbed. 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 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 access whether the channel is already in use at the time. If positive, it backs off, going to the 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 collision in an ALOHA-like protocol (Figure 5). They also evaluated the impact of these method on energy consumption, showing that the contention based strategies actually achieve a better energy performance as the networks become denser.

![Packet Error Rate for different medium access protocols. Reproduced from [18].](image)

Despite this, the module available in [18] lacks their implementation of CSMA and CSMA-x, as well as any sample code on how to interface with the energy framework. Furthermore, no documentation is provided, and the code is very poorly commented. This makes Module IV very hard to work with.

IV. Module Installation and Testing

In this section, we describe our experiences when testing the four modules. Installations scripts are provided in [33].

A. Module I

Installation of Module I is accomplished by cloning the repository [20] 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 describes 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 understating how the code is structured. Additionally, the authors created a chat in [34], where anyone can ask a question about the module’s usage.

The code itself is well organized, and most of the classes have some commentary explaining their function. The module closely follows ns-3 coding style [35], using “helpers” objects to setup and configure the network. This makes Module I the most user-friendly of all implementations. Module I comes with three example files showcasing how to setup a simulation.

The first, `simple-lorawan-network-example.cc`, illustrates what steps are made 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 to generate data. Finally, the simulation is started.

While very simple, since a network composed of a single ED and GW is created, this example is very useful to understand what classes are involved in the process of generating and sending a packet. Logging is enabled by default on the example, meaning 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.

The second example, `complete-lorawan-network-example.cc`, showcases how to configure a more complex network, with thousands of EDs. While the commentary claims the example can create a network with tens of GWs, there is some missing code to make this work. The simulation will create only one GW, in spite of the value assigned to the `gatewayRings` variable. One of the missing functions, `HexGridPositionAllocator`, would assign the positions of gateways so that they are placed in a hexagonal grid layout. This could be useful to simulate a dense IoT urban scenario, where dozens of GWs are placed in order to cover a large area. The author states in [34] that while the framework for multiple GWs is done, the code to collect and interpret data from multiple GWs is still in development.

Nevertheless, the example will work normally if the user creates a single GW. EDs will be placed randomly in a circle of a defined radius, and each ED will be assigned a SF based on the received power level computed by the `link measurement model` described previously. The SF assigned is the smallest one so that the ED transmissions can reach the GW above the GW’s sensitivity. A channel model considering only path loss is instantiated. The EDs are setup with an application that periodically sends messages, and the simulation is started.

The metrics examined are the packet delivery ratio, and the ratio of packets lost due to interference, lack of open reception channels or under-sensitivity. A file with the position of every ED and their assigned SF is also generated. Figure 6 shows a graphic of one simulation with 2000 EDs, positioned randomly in a 15 km circle centered on the GW. Its possible to see that the farther an ED is from the GW, the higher its SF is. This is because those EDs experience a greater path loss, and so need to use a bigger SF to allow their signals to reach the GW within the sensitivity threshold. The module also provides a
gnuplot script to create plots like the Figure 6, though it will not work because of a missing file. This can be easily corrected by eliminating the lines related to the buildings.dat file.

Fig. 6. SF allocation for a network of 2000 EDs and a central GW.

Despite these problems, the example is useful for showcasing how to build a more complex network and how to obtain certain metrics. The final example is network-server-example.cc. It creates a network composed of an ED, a GW and a NS, and showcases how to setup these components to interact with each other. It is similar to the first example in that it showcases the step-by-step process that the simulation takes to generate and transmit a packet through the ED to the GW, and how the NS interprets it.

B. Module II

Module II installation is accomplished in the same way as the previous module. The repository is found in [26].

Unlike Module I, there is no documentation explaining the main features of the module. The article that introduces the module serves as the best reference to understand the implementation. Besides that, the code is well organized and comply to ns-3 standards.

The only example provided is lora_battery.cc. The example 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 number of re-transmissions. Overall, it is a good sample code to understand how to use the module.

C. Module III

Module III is available to download in [29]. Unlike the other modules, this repository contains the entire ns-3 directory with the module inside. This makes it awkward to port the module to different ns-3 installations, though we verified that it is possible to do so.

The module includes a short text file with a brief explanation of the module. The module comes with many example codes. However, most of those are broken and will cause ns-3 fails to build them. 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 built.

The authors state in their article [30] that this example was used for all simulations in their work. The code is indeed very extensive, and seems to show almost all of the features of the module. This makes it very hard to understand the code, especially considering its poor 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 generates 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 uses compilers incompatible with C++11. Version 5.0 or higher of gcc is required to run the example.

D. Module IV

To install Module IV, the repository in [32] must be cloned to the src directory of the ns-3 installation. Before ns-3 configure and build commands are executed, the folder must be renamed to lora. This is because the name of the repository does not match the name given to the module in the build scripts.

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

The code has very poor documentation, making it hard to understand the process by which the simulation is setup. A series of methods that are defined in the example are called, instead of using the approach of helpers to configure the network. The module itself also lacks commentary and documentation. Finally, the classes related to CSMA and the energy framework are missing from the module.

V. CONCLUSION

In this paper, we presented a overview of the available tools for simulation of LoRaWAN in ns-3. The development of those simulators is important to aid the community to better understand and assess the protocol. Table IV presents a overview of the four modules.

In our survey, we found Module I to be the easiest to work with, as very good documentation and organized code are provided. Thought many features are not available yet, such as multiple gateway simulation and energy framework
integration, the authors have shown they are still working on the module, and provide an 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 of 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 implementation of algorithms on the server side, such as ADR schemes. However, its documentation is very scarce.

Module III also provides a lot of features not available in the first module, like the possibility of simulating multiple gateways and ACK messages. Its original paper [30] provides a detailed description of the module, thought the code in the repository requires some work to compile correctly, and the sample code provide is very unorganized.

Finally, Module IV provides the only one validated through real measurements. However, missing functions and lack of documentation, as well as a broken sample code makes it hard to work with.

Overall, we believe Module I is the most promising, considering the continuous support of the authors and popularity in other researches. However, if multiple gateways are necessary, or one wishes to evaluate energy consumption or inter-protocol interference, Module II and III are preferred. In addition, Module II is the better choice for evaluating server-side algorithms, as its NS has the most complete implementation of all modules.

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 Tutorials, vol. 17, no. 4, pp. 2347–2376, 2015.

[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] Federico Montori and Luca Bedogni and Marco Di Felice and Luciano Bononi, “Machine-to-machine wireless communication technologies for the Internet of Things: Taxonomy, comparison and open issues,” Pervasive and Mobile Computing, vol. 50, pp. 56 – 81, 2018.

[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.

[5] J. Petajajarvi, K. Mikhaylov, A. Roivainen, T. Hanninen, and M. Petissalo, “On the coverage of LPWANs: range evaluation and channel attenuation model for LoRa technology,” in 2015 14th International Conference on ITS Telecommunications (ITST), 2015, pp. 55–59.

[6] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, “Understanding the Limits of LoRaWAN,” IEEE Communications Magazine, vol. 55, no. 9, pp. 34–40, 2017.

[7] M. Bor, U. Roedig, T. Voigt, and J. M. Alonso, “Do LoRa Low-Power Wide-Area Networks Scale?” 11 2016.

[8] M. Kahir, S. Islam, M. Hossain, and S. Hossain, “Detail Comparison of Network Simulators,” International Journal of Scientific and Engineering Research, vol. 5, p. 203, 11 2014.

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

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

[11] J.-M. G. Claire Goursaud. (2015) Dedicated networks for IoT: PHY / MAC state of the art and challenges. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01231221

[12] K. Staniec and M. Kowal, “LoRa Performance under Variable Interference and Heavy-Multipath Conditions,” Wireless Communications and Mobile Computing, 2018.

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

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

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

[16] B. Reynenders, Q. Wang, and S. Pollin, “A LoRaWAN module for ns-3: implementation and evaluation,” 2018.

[17] T. C. A. Augustin; J. Yi and W. Townsley, “A Study of LoRa: Range & Low Power Networks for the Internet of Things,” Sensors, September 2016. [Online]. Available: https://www.mdpi.com/1424-8220/16/9/1466

[18] A. Duda and T.-H. To, “Simulation of LoRa in NS-3: Improving LoRa Performance with CSMA,” in IEEE ICC, Kansas City, United States, May 2018. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01835883

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

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

[21] D. M. C. L. Vangelista, “Performance Evaluation of LoRa Networks in a Smart City Scenario,” IEEE ICC, 2017.

[22] M. C. D. M. A. Zanella, “Confirmed Traffic in LoRaWAN: Pitfalls and Countermeasures,” 17th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net), 2018.

[23] Ouksoussou Yassine; Mohamed Baslam and Mohamed Ouksoussou, “LPWAN IEEE 802.11ah and LoRaWAN capacity simulation analysis comparison using NS-3,” 4th International Conference on Optimization and Applications (ICOA)), 2018.

[24] Kouvelas, Nikos and Rao, Vijay and R. Venkatesha Prasad, R, “Employing p-CSMA on a LoRa Network Simulator,” 2018.
[25] Michele Luvisotto, Federico Tramarin, Lorenzo Vangelista, and Stefano Vitturi, “On the Use of LoRaWAN for Indoor Industrial IoT Applications,” *Wireless Communications and Mobile Computing*, 2018.

[26] (2017) ns3 lora module. [Online]. Available: https://github.com/networkedsystems/lora-ns3

[27] 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*, vol. 5, no. 3, pp. 1830–1842, 2018.

[28] B. Reynders, W. Meert, and S. Pollin, “Power and spreading factor control in low power wide area networks,” in 2017 *IEEE International Conference on Communications (ICC)*, May 2017, pp. 1–6.

[29] 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

[30] F. V. den Abeele, J. Haxhibeqiri, I. Moerman, and J. Hoebeke, “Scalability Analysis of Large-Scale LoRaWAN Networks in ns-3,” *IEEE Internet of Things Journal*, vol. 4, no. 6, Dec 2017.

[31] 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 Mobile Computing Conference (IWCMC)*, 2018.

[32] D. M. M. C. R. M. Luvisotto. (2017) LoRaWAN ns-3 module. [Online]. Available: https://github.com/drakkar-lig/lora-ns3-module

[33] (2018) LoRaWAN ns-3 modules installation scripts. [Online]. Available: https://github.com/dflor

[34] Module I gitter char. [Online]. Available: https://gitter.im/ns-3-lorawan

[35] ns-3 Coding Style Guidelines. [Online]. Available: https://www.nsnam.org/develop/contributing-code