Improving the FLoRa Simulation Framework for the Performance Evaluation of IoT Scenarios

Jose-Manuel Martinez-Caro†, Maria-Dolores Cano
Department of Information Technologies and Communication
Universidad Politécnica de Cartagena
Cartagena, Spain
email: {josem.martinezcaro, mdolores.cano}@upct.es

Abstract— **In the last years, Low-Power Wide Area Network (LPWAN) technologies have increased their presence. Their main characteristic is covering large areas with limited resources. One of the greatest exponents in LPWAN is the entire system composed by Long-Range (LoRa) and (LoRaWAN). LoRa offers an easy deployment, a low-power consumption, a wide-coverage, and a high performance, although it also has several constraints such as a low Data-Rate (DR), a duty-cycle restriction (1%), and a limited application for real-time services. In this paper, we improve the Framework for LoRa (FLoRa) network simulation framework using open-source tools and the programming languages C++ and Python. The incorporated options allow a better adaptation of the simulation to users' requirements (topology, network conditions, or typical LoRa setting parameters, such as Spreading Factor (SF), Transmission Power (TP), Coding Rate (CR), Bandwidth (BW), or Carrier Frequency (CF), among others). As an example, we show the performance of a simulated air-quality monitoring system deployed using LoRa/LoRaWAN with a real dataset. System performance is evaluated in terms of several quality metrics. By using simulation tools like the one we present in this work, IoT (Internet of Things) networks and services can be tested and evaluated in advance, facilitating a better planning of future real deployments.**

Keywords- IoT; LPWAN; LoRa; simulation; OMNeT++; FLoRa Framework.

I. INTRODUCTION

The Internet of Things (IoT) concept is described as a dense, large-scale, open and dynamic ecosystem of social-technical entities and applications [1]. This recent concept has shaken the network up with new devices and systems, building a more heterogeneous network, where the interconnection between devices and systems to transmit and receive data is simpler [2]. These data will be further processed to provide information and make decisions. Moreover, IoT will allow an exponential increase of connected devices to the network, rising to almost 31 billion by 2020 and more than 75 billion by 2025 [3].. It will suppose an enormous economic injection to the technology market [3]. IoT is present in many application fields, such as Smart-Home, Smart-City, Industry 4.0, Smart-Grid, etc. [4]. This technology is still in a research and development phase, in spite of the forthcoming massive deployment in short-medium term [5].

IoT features perfectly match with Low-Power Wireless-Area-Network (LPWAN) technologies, which stand out for their resource efficiency, i.e., low-power transmission [6]. IoT and LPWAN share several features such as low-cost, low-power, high-performance, wide-coverage, low Data-Rate (DR), and fast-arrangement, where dense-device deployments could be done in a determined area connected to one or multiple gateways. Over rural areas, LPWAN could achieve communication distances around 30 kilometers between emitter and receptor [7], which means a great enhancement compared to Wireless Local Area Network (WLAN) [8] or Wireless Sensor Networks (WSN) [9]. In addition, LPWAN technologies consume less energy in comparison with cellular networks (2G, 3G, 4G). On the other hand, LPWAN is not suitable for all services and operations because it only sends light and infrequent frames given the limited data rate imposed to fulfill the duty cycle restriction which must not exceed 1% of the time over the Industrial, Scientific and Medical (IMS) band (EU: 868MHz and 433MHz; USA: 915MHz and 433MHz). LoRa/LoRaWAN [10], Weightless [11], NWave [12], Telensa [13], Random Phase Multiple Acces (RPMA) [14], Sigfox [15], and Narrow Band-IoT (NB-IoT) [16] are some of the multiple examples of LPWAN.

One of the LPWAN technologies with a higher popularity is LoRa/LoRaWAN due to its performance. In this work, we present the improvements done in a simulation software to evaluate the performance of LoRa/LoRaWAN networks and services. The modifications are done using different libraries and frameworks at low-level. The base of the simulation tool is the Framework for LoRa (FLoRa) [17] and OMNeT++ [18]. Specifically, our contributions are:

1. FLoRa implements a simplified version of the Okumura-Hata model. However, it is not accurate because it is an approximation based on linear regression whose outcomes do not match the results obtained in the related scientific literature. We introduce a more precise implementation of the Okumura-Hata wireless propagation model.
2. Automatic assignment of some LoRa parameters for static LoRa nodes. We propose a new simple algorithm to automatically set Spreading Factor (SF) and Transmission Power (TP) for each LoRa node according to the LoRa end-nodes location in relation to the LoRa gateway (LoRaGW) position.
3. Introduction of security mechanisms in LoRa. We include encryption/decryption and digital signature in the

communication using Counter Mode (CTR) and the Cipher-based Authentication Code (CMAC) mechanisms, respectively. Both methods are based on Advanced Encryption Standard (AES).

4. Performance evaluation tasks. Using our new implementation, it is possible to change the simulation environment and automatically adapt the parameters to this simulation. After a sample period (Teval), the simulator computes several quality components from numerous quality metrics, namely, Quality of Data (QoD), Quality of Information (QoI), Quality of user Experience (QoE), and Quality-Cost (QC). As an example, we briefly present the performance of a simulated air-quality monitoring system deployed using LoRa/LoRaWAN with a real dataset. An example will be shown under a rural environment, allowing an efficient performance evaluation.

The rest of the paper is organized as follows. In Section 2, we briefly describe LoRa/LoRaWAN and report an overview of the state-of-the-art in computer simulation tools for LoRa. Section 3 describes the software used as the basis for this work. Section 4 details the improvements that we have incorporated and their advantages, with an example of the performance evaluation outcomes. The paper ends summarizing the most important results of this work.

II. Related Works

A. LoRa/LoRaWAN

In a typical LoRa/LoRaWAN deployment (see Figure 1), there are three main devices: LoRa end-nodes, which acquire data from sensors at the application layer (from a simulation perspective) and send these data using LoRa physical layer; one or more LoRaGW that receive LoRa frames and cast them to be forwarded through a wired network; and one or more Network Servers, usually in the cloud, which will process the received data and are likely in charge of decision-making.

LoRa physical-layer uses Chirp Spread Spectrum (CSS) modulation over the Industrial, Scientific and Medical (ISM) frequency band which varies according to the region. Europe uses 868MHz whilst USA adopts 915MHz, though 433MHz is common in both regions. To gain resilience to interference and noise, LoRa spreads a narrowband signal over a wider channel bandwidth [4] and the sensibility of the receiver is 19.5 dB below the noise floor. There are multiple parameters that characterize LoRa communication between LoRa end-nodes and LoRaGW: Spreading Factor (SF), Transmission Power (TP), Carrier Frequency (CF), Coding Rate (CR), and Bandwidth (BW). First, SF varies from 7 to 12 (both included). SF define the coverage area, where higher SF values achieve higher ranges but with lower Data-Rate (DR). Second, TP ranges theoretically from -4dBm to 20dBm. It sets the intensity that LoRa end-nodes use to transmit LoRa data frames to the LoRaGW. Observe that the higher SF and TP, the larger the coverage area. Third, CF is the middle frequency in steps of 61Hz within the range according to the region. Fourth, CR provides security against interferences, where higher values provide higher protection (4/5, 4/6, 4/7 and 4/8) [19]. BW is the frequency width in the transmission band and the wider BW is, the higher DR, though sensibility is lower. Lastly, Time on Air (ToA) is the time to transmit a frame from a LoRa end-node to the LoRaGW and depends on SF and BW, being opposite to the DR parameter. The technology has three degrees of diversity (time, frequency, and SF) [4]. The communication between LoRa end-nodes and LoRaGW can be unidirectional or bidirectional. Unicast, multicast, and broadcast are the three types of communication addressing available in LoRa networks. The duty-cycle is limited and should be lower than 1% of the time, having a high repercussion on the maximum transfer-rate. Depending on the application, this constraint makes this technology inappropriate for many services that require constant data transmission. Some authors propose the implementation of algorithms such as Adaptive Data-Rate (ADR) [20], Distributed Coordination Functions (DCF) particularly Carrier Sense Multiple Access (CSMA) [21], and Channel Activity Detection (CAD) [22][23], with the aim of managing link parameters and getting adequate network processes, providing medium access control mechanisms as CSMA and detecting the LoRa preamble on the channel with maximum power efficiency, respectively.

On the other hand, LoRaWAN [10] specifies the architecture, layers, and protocols operating over LoRa. Mesh or star are the two possible network architectures. There are three LoRa end-node classes (A, B and C), all classes observing the duty-cycle. Class A may open a collecting window to receive acknowledgments or new messages after a specific time lapse. Class B adds scheduled received windows to class A and class C keeps the receive window open at any time. The Network Server deletes duplicate packets if multiples gateways are deployed and redirect the packet to the corresponding Network Server. If the application servers exist, then the Network Server will send the information to them.

B. Simulation tools for LoRa/LoRaWAN

Computer simulators are complete tools to replicate real network operation without the need of acquiring hardware, but programming skills are required to define simulation conditions with a blow of code. Simulators are also useful to test large networks with hundreds or thousands of devices on the network that are too costly in time (by placing and programing) and expenses [24]. For instance, the study done in [25] models the LoRa network efficiency and demonstrates the exponential increase of packet drops with the raising of devices due to interferences in a small area and LoRaWAN access methods. Some well-known network simulators are OMNeT++, NS3 [26], NetSim [27], SimPy [28], and OPNET [29], among others. Moreover, multiple frameworks and libraries are available to be imported such as FLoRa Framework. FLoRa allows recreating a LoRa network scenario under desirable conditions using the OMNeT++ simulator and the INET framework. More details about FLoRa are given in the next section.

Cooja framework is another simulation tool that runs programs to simulate and evaluate the performance of different networks, such as WSN or IoT-based projects such

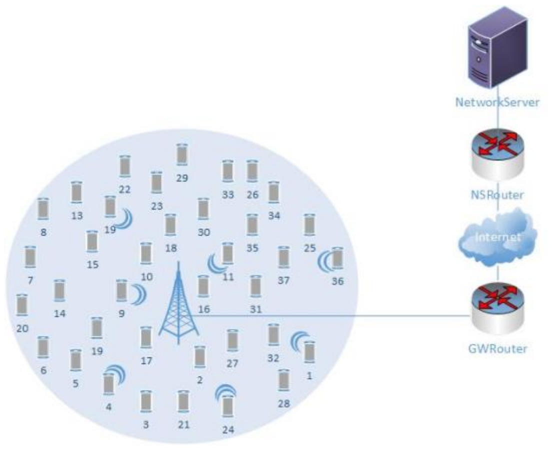

Figure 1. LoRa network and backbone network.

as LoRa using Cooja [24]; it is not a specific framework for LoRa networks though. One of the strengths of Cooja is its simulations taking into account the devices' energy consumption. Cooja is open-source and uses C programming language. It includes low-power protocols to define the simulation settings and uses SimPy to carry out the simulation enabling a graphical interface.

On the other hand, LoRaSim is a discrete event simulator implemented in a 2D scenario. It also uses SimPy to place LoRa end-nodes and LoRa sinks. This framework sets the LoRa parameters described previously and packets payload. LoRaSim uses Semtech SX1301 as the LoRa reference module compatible with Semtech SX1272/SX1276 (used in the FLoRa Framework), which is able to receive 8 concurrent orthogonal signals. Additionally, LoRaSim includes two evaluation metrics: Data Extraction Rate (DER), the ratio of received to transmitted messages over a period of time and Network Energy Consumption (NEC), as the energy spent by the LoRa end-node to successfully achieve the LoRa sink. NEC depends mainly on the transceiver state and time per state, and it should be minimum to extend as much as possible the batteries life of the devices. LoRaSim is open-source and requires additional libraries, e.g., Matplotlib, SimPy, and Numpy, but it has not a graphical interface. Finally, it includes several examples and low-power protocols implementations. Table I summarizes the most important characteristics of the simulation tools for LoRa environments.

III. BASELINE SOFTWARE

We use open-source tools available online to implement the simulator. These resources allow the user to develop a complete LoRa network simulation environment adaptive to any required scenario, getting an exhaustive performance evaluation of the designed topology. The baseline tools used in this work are: OMNeT++, INET Framework, FLoRa Framework, and Crypto++ [30].

A. *OMNeT++*

OMNeT++ IDE uses Eclipse [31] as the main developer platform and enhances it with new functions such as new editors, views, wizards, and so on. It allows users to create new and/or re-configure existing models using Network Description (NED) language, and configuration files (.ini). Then, the simulator evaluates the performance taking into account the obtained results. All of this is using C++ programming language, git integration, and other open-source tools and components. NED files define and edit the model graphically or by text. Both options are able to create compound modules, channels, and other component classes, as well as other object features. On the other hand, the ini file provides the parameters to adapt and configure models to the simulation, and as the NED files, is edited graphically or by text. An ini file recognizes all NED components from the top-level module to the last inherited module, being possible to define new parameters different to the default ones in all existing modules. Moreover, ini files enable users to define different scenarios according to the set parameters or random number seed.

More than one process can be run at the same time, so the building process is faster. While the simulation process runs in a new window, the user can continue developing the program due to this parallel operation. Once the simulator has finished, the results are preserved into a vector (as a collection of all intermediate results) and scalar files. The default Integrated Development Environment (IDE) or other external tools (e.g. Python) are available to the analysis of the results.

B. *INET Framework*

New frameworks can be added to OMNeT++ to provide new capabilities to the simulator. Particularly, INET includes agents, protocols, and many other models to create, redefine, or certify new protocols or scenarios. The supplied models in INET are for physical, link, network, transport, and application communication layers for different types of communication networks such as wired, wireless, ad-hoc, or WSN. INET bases its operation on message exchanges between modules.

TABLE I. COMPARISON OF LORA SIMULATION TOOLS

Features	**Simulation Tools**		
	FLoRa Framework	***Cooja Framework***	***LoRaSim***
Base Simulator	OMNeT++	Contiki OS RIOT OS	Python
Programming Language	C++	C	Python
Additional Frameworks	INET	SimPy	Matplotlib SimPy Numpy
Graphical Interface	Yes	Yes	No
Software Licence	Open-source	Open-source	Open-source
Power Awareness	Yes	Yes	Yes
Low-Power protocols	Yes	Yes	Yes
Examples	Yes	Yes	Yes
Last Version	0.8	3.0	0.2.1

C. FLoRa Framework

As INET, FLoRa is a specific framework to test LoRa/LoRaWAN networks. It enables physical and link layer evaluation, defining one (or more) gateways in the network where end-nodes will send data frames to, supporting bi-directional communication, defining the path for messages from source to destination (LoRa end-nodes to Network Server), and estimating the energy consumed by LoRa end-devices. FLoRa sets the main LoRa/LoRaWAN parameters, namely, SF, CF, BW, CR, and TP, which influence the communication coverage and the probability of data frames collision. As LoRa transmission uses the wireless interface, a frame is received correctly if the received power (which depends mainly on SF and TP) is higher than the sensitivity of the LoRaGW. The framework also estimates the energy consumption of each LoRa node according to both the time spent by the LoRa radio module in a specific state (transmit, receive, sleep, and off) and the TP value. Semtech SX1272/73 datasheet provides the consumptions for each state with a supply voltage of 3.3V.

Lastly, a typical deployment is not usually only composed of LoRa end-nodes and LoRaGW. As an example, we usually include in our simulations a backbone network behind the LoRaGW to reach a Network Server (Figure 1). In our case, once the LoRaGW receives a LoRa frame, it encapsulates the frame into an EthernetIIFrame and forwards it to the Network Server using the TCP/IP protocol stack, particularly, User Datagram Protocol (UDP) messages. This part is mainly simulated using the INET modules explained previously. Network Server will discard duplicate packets if the same packet is received by multiple LoRaGWs.

D. Crypto++

Crypto++ is an open-source library based on C++ programming language that includes algorithms for ciphering, message authentication codes, hash generators, public-key cryptosystems, etc. Crypto++ implements multiples methods and schemes such as Diffie-Hellman, Advanced Encryption Standard (AES), RSA, Elliptic Curve Cryptography (ECC), and Digital Signature Algorithm (DSA), among others [30].

IV. Novel Incorporated Tools

This section describes the improvements and modifications that we have incorporated into the simulation software, with the aim of having available an easy to use performance evaluation tool for IoT services and networks based on LoRa/LoRaWAN.

A. Wireless Propagation Model

The FLoRa framework includes an Okumura-Hata implementation. From the FLoRa documentation, it is known that this implemented wireless propagation model is based on an approximation, using a linear regression with three factors, namely, K1, K2, and the distance between a LoRa end-node and the LoRaGW. The first two factors, K1 and K2, take the default values of 127.5 and 35.2, respectively. However, this method is not precise to estimate the Free-Space Path Loss (FSPL), since it reaches a maximum distance of around 6 km (as observed in extensive simulations). It can be verified in the related literature that this distance is too small for this technology [7][32]. Additionally, with this implementation of the Okumura-Hata model, it is not possible to choose one of the three available environments that the original Okumura-Hata provides, namely, rural, sub-urban or urban.

Consequently, we introduce a new Okumura-Hata model implementation in FLoRa to accurately estimate the FSPL in the simulator, taking into account those three possible environments (rural, sub-urban, and urban). FSPL is lower in rural scenarios and higher in urban environments, because in the former there might not be buildings that interfere with the electromagnetic wave propagation, contrary to the urban environment. The main objective of using this type of propagation models is to represent properly the effect of the physical layer in the simulations, providing an environment as real as possible and discarding the use of less accurate regression methods. This new model implementation is defined by (1) [33] and uses more rigorous factors such as frequency (f), the distance between a LoRa end-node and the LoRaGW (d), LoRa end-node height (hm), and LoRaGW height (hb) [33].

$$\begin{aligned} a(h_m) &= 3.2(\log_{10}(11.75 \cdot h_m)^2 - 4.97 \\ L_{urban} &= 69.55 + 26.16\log_{10}(f) - 13.82\log_{10}(h_b) - a(h_m) \\ &\quad + (44.9 - 6.55\log_{10}(h_b)) \cdot \log_{10}(d_m) \\ L_{sub\text{-}urban} &= L_{urban} - 2(\log_{10}(f/28))^2 - 5.4 \\ L_{rural} &= L_{urban} - 4.78(\log_{10}(f))^2 + 18.33\log_{10}(f) - 40.94 \end{aligned} \tag{1}$$

B. Initial settings parameters in LoRa

Additionally, we use Python: a general-purpose programming language to generate automatically an ini file that sets the configuration parameters according to the desired conditions, for instance: the environment (rural, suburban, urban), number of LoRa end-nodes, or performance evaluation period (Teval), among others. Every Teval, the simulation tool will show the calculated performance quality metrics.

In the same script, we define the automatic selection of SF and TP values for each LoRa end-node. These values will depend on two factors: the distance between the LoRa end-node and the LoRaGW and the distance between the farthest LoRa end-node and the LoRaGW. Note that we are working with fixed LoRa nodes and future improvements will be added for mobile nodes. Given that to set SF we have 6 possible options [SF7, SF12], we assume that there are 6 possible distance intervals from 0 until the maximum (farthest) distance, i.e., the LoRa end node that is farther from the LoRaGW. Depending on what interval fits the distance from LoRa end-node i to the LoRaGW, the algorithm assigns the corresponding SF, knowing that lower SF values are used for LoRa end-nodes closer to the LoRaGW, and vice versa. Likewise, we follow the same method to select the appropriate value for the TP, but using

12 possible intervals for 12 possible values [2dB,14dB], assigning lower values to LoRa end-nodes nearer to the LoRaGW.

According to the European case, SF and BW combination results in 7 different DR [DR0, DR6], each one with a specified Maximum Payload Size (M). With all these parameters it is possible to compute the maximum Time-on-Air (ToA) for each DR class impacting in the effective throughput due to data-cycle (1%). We calculate the ToA and the minimum time between packets using a payload of 12 bytes and a duty-cycle of 1% (see Table II).

After the simulation, the Python script processes the results and represents them graphically for a better user comprehension and to facilitate the computation with other environments.

C. *Security*

We have modified the simulation tool so that transmitters and receivers use the AES Counter Mode (CTR) method to encrypt and decrypt messages. CTR is a symmetric encryption method, so it employs a shared private key to encrypt/decrypt messages, whose content will be hidden while flowing through the network. That is, the encrypted message is sent through the wireless channel and it can be only decrypted by those recipients sharing the same private key.

Taking the same shared key (or a different one but also shared), the message is signed using the AES Cipher-based Message Authentication Code (CMAC) method. This digital signature guarantees authentication (the origin of the message is verified) and integrity (the data has not been modified or altered along the communication path). These processes (cipher/decipher and sign) are implemented with Crypto++ libraries imported into OMNeT++. In both algorithms, the key length is 128 bits.

D. *LoRa network*

By default, SimpleLoRaApp is the application module in a LoRa end-node in FLoRa (see Figure 2). This module generates a random number to schedule the messages transmission (e.g., following an exponential distribution) and sends the message to the LoRa physical layer, which is responsible for transmitting the message in plain text. Then, the LoRa end-node sends a RadioFrame to LoRaGW that contains a LoRaAppMessage encapsulated in it.

To compute the number of lost packets and measure the Packet Delivery Rate (PDR) in the LoRa/LoRaWAN, we modify the radio interface of LoRaGW as follows. When LoRaGW receives a new packet, it checks two values: its sequence number and its source ID (LoRa end-node ID). If for a source ID i the received and expected sequence number match, the number of lost packets is 0. In contrast, if for a source ID i the sequence number received is higher than expected, the difference should be the number of lost packets. Since we compute new intermediate metrics in different modules and OMNeT operations is based on message exchange, LoRaAppMessage and LoRaMacFrame modify their payload to carry out the information to the Network Server.

Considering the simplicity of the LoRa end-node application in FLoRa, we redefine it with three submodules with specific functionalities, namely, Read, CipherData, and SimpleLoRaApp, as depicted in Figure 2. The Read module allows each LoRa end-node to read from its own dataset (e.g., from a real one as it will be shown later). Once LoRa end-node acquires the data, it sends a ReadDataPacket message to the CipherData module. This module receives read data and initializes the symmetric encryption process, sending the data to the next module (SimpleLoRaApp) encrypted and signed as explained before. The last module is SimpleLoRaApp, which passes the message to the LoRa physical layer and sends it via the wireless channel. It is important to note that before the physical layer receives the frame, we have also added two throughput meters. The goal is to know the generated and received traffic by each LoRa end-node (bits/s and packets/s).

E. *Backbone Network*

When the LoRaGW receives a new message, it encapsulates the new message into a EthernetIIFrame and forwards it to the Network Server using UDP. This transport protocol implements message delivery through the network with a simple connectionless communication model, without confirmation or flow control. In its original form, the Network Server is defined as StandardHost and when messages arrive to the application module (called udpApp[0]) its only purpose is to count the number of received messages, discarding the message content. The Network Server does not check if the message is duplicated or not, which is insufficient for our needs (Figure 2).

Therefore, we have modified the Network Server splitting it into three independent modules, namely, CommunicationParameters, Decrypt, and Processing. First, CommunicationParameters computes metrics to quantify quality components at different abstraction levels.

TABLE II. LoRa Parameters according to SF y BW

DR	SF	BW (kHz)	M (bits)	Throughput (bps)	ToA (ms)	Min time between Packets (s)
0	12	125	59	250	51	148.3
1	11	125	59	440	51	82.3
2	10	125	59	980	51	41.2
3	9	125	123	1760	115	20.6
4	8	125	250	3125	242	11.3
5	7	125	250	5470	242	6.2
6	7	250	250	11000	242	3.1

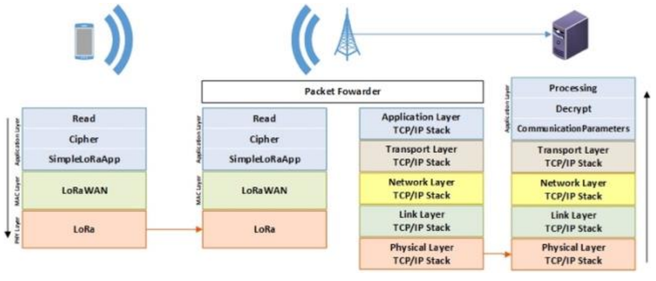

Figure 2. LoRa and backbone network protocol stack in our improved simulation tool.

Particularly, we measure metrics such as precision, accuracy, timeliness, delay, jitter, throughput, PDR, energy consumption, etc., from which we derive QoD, QoI, QoE, and QC. Using these for quality components, the performance of a service based on IoT and LoRa/LoRaWAN can be easily evaluated [34]. The received message and these metrics are sent to the Decrypt module that obtains the plain text from the ciphered text using the shared key and also checks the message signature. In case of a wrong signature, the message is discarded. The last module is Processing, which carries out two functions. First, storing the metrics for each received packet in a T_{eval} and second, when timer (T_{eval}) is over, the simulator processes the metrics and computes the quality components using the received metrics during that period.

F. Use case example

Our example is based on a real air-quality monitoring system, whose dataset measurements will be used in the simulations. The dataset [35] is part of a group of air-quality stations that take multiple measures (Humidity, Temperature, Pressure, CO, NO, etc.), located in a Spanish region. The dataset is preprocessed, splitting it into different files, one per LoRa end-node.

The simulated scenario is composed 53 LoRa end nodes, one LoRaGW, and one Network Server. The location of LoRa end-nodes is set according to the chosen environment (having a higher area of rural environment) and the LoRaGW is located in the center of all LoRa end-nodes. From all the obtained metrics, quality components are derived and normalized for a better comparison (Figure 3), so that the closer to 1 the better the performance. The method to obtain QoD, QoI, QoE, and QC is described in [34][36] and it is out of the scope of this paper.

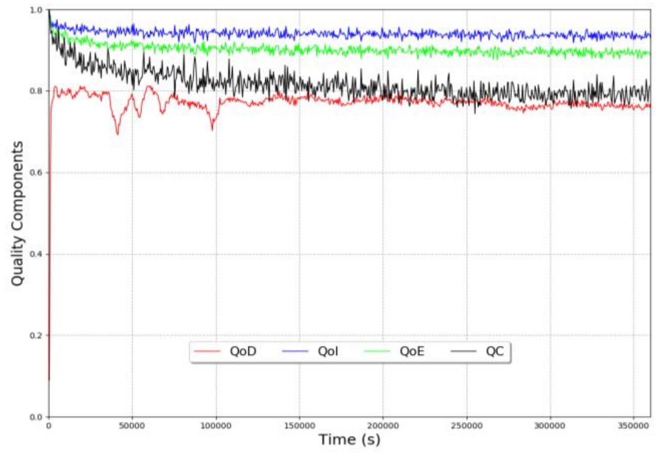

Figure 3. Performance of the LoRa-based air-quality monitoring system in terms of several quality components.

V. CONCLUSIONS

The influence that IoT-based services may have on social and industrial scenarios requires validating the proposed schemes before a real deployment is done. We presented in this paper a LoRa network simulator environment to study the performance of new applications and services under specific conditions, using as a baseline different libraries and frameworks available. Particularly, we used FLoRa and OMNeT, and introduced new features and several modifications. Among others, a new Okumura-Hata model has been implemented improving the accuracy of the simulation tool, and new modules to read, cipher, and send data from the emitter to the receiver have been incorporated. As a future work, we plan to introduce new methods to improve performance evaluation and monitoring based on advanced quality metrics.

ACKNOWLEDGMENT

This work was supported by the AEI/FEDER-UE project grant TEC2016-76465.C2-1-R (AIM).

REFERENCES

[1] G. Fortino, C. Savaglio, and M. Zhou, "Toward opportunistic services for the industrial Internet of Things," *IEEE Int. Conf. Autom. Sci. Eng.*, vol. 2017-Augus, pp. 825–830, 2018, doi:10.1109/COASE.2017.8256205, ISBN: 9781509067800, ISSN:21618089.

[2] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Commun. Surv. Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015, doi:10.1109/COMST.2015.2444095, ISSN:1553877X.

[3] "Internet of Things (IoT), Connected Devices Installed Base Worldwide From 2015 To 2025," 2015. [Online]. Available: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/. [Accessed: 22-Oct-2019].

[4] U. Raza, P. Kulkarni, and M. Sooriyabandara, "Low Power Wide Area Networks: An Overview," *IEEE Commun. Surv. Tutorials*, vol. 19, no. 2, pp. 855–873, 2017, doi:10.1109/COMST.2017.2652320, ISSN:1553877X.

[5] R. Casadei, G. Fortino, D. Pianini, W. Russo, C. Savaglio, and M. Viroli, "Modelling and simulation of Opportunistic IoT Services with Aggregate Computing," *Futur. Gener. Comput. Syst.*, vol. 91, pp. 252–262, Feb. 2019, doi:10.1016/j.future.2018.09.005, ISSN:0167739X.

[6] R. Sanchez-Iborra and M. D. M.-D. Cano, "State of the art in LP-WAN solutions for industrial IoT services," *Sensors (Switzerland)*, vol. 16, no. 5, 2016, doi:10.3390/s16050708, ISBN: 978-3-03842-370-6, ISSN:14248220.

[7] 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," *2015 14th Int. Conf. ITS Telecommun. ITST 2015*, pp. 55–59, 2016, doi:10.1109/ITST.2015.7377400, ISBN: 9781467393829.

[8] S. Chieochan, E. Hossain, and J. Diamond, "Channel assignment schemes for infrastructure-based 802.11 WLANs: A survey," *IEEE Commun. Surv. Tutorials*, vol. 12, no. 1, pp. 124–136, 2010, doi:10.1109/SURV.2010.020110.00047, ISSN:1553877X.

[9] L. M. Borges, F. J. Velez, and A. S. Lebres, "Survey on the characterization and classification of wireless sensor network applications," *IEEE Commun. Surv. Tutorials*, vol. 16, no. 4, pp. 1860–1890, 2014, doi:10.1109/COMST.2014.2320073, ISSN:1553877X.

[10] LoRa Alliance, "LoRaWAN - What is it?. A technical overview of LoRa and LoRaWAN," no. November, pp. 1–20, 2015.

[11] "Weightless.," 2019. [Online]. Available: http://www.weightless.org/.

[12] "NWAVE Technology.," 2019. [Online]. Available: https://www.nwave.io/.

[13] "Telensa.," 2019. [Online]. Available: https://www.telensa.com/technology.

[14] Ingenu, "RPMA Technology," 2019. [Online]. Available: https://www.ingenu.com/technology/rpma/.

[15] "Sigfox.," 2019. [Online]. Available: https://www.sigfox.com/en. [Accessed: 09-Sep-2019].

[16] Qualcomm Incorporated, "RP-151621- Narrowband IOT," 2015.

[17] "FLoRa Framework," 2019. [Online]. Available: https://flora.aalto.fi/. [Accessed: 05-Feb-2019].

[18] "OMNeT++ Simulator." [Online]. Available: https://omnetpp.org/. [Accessed: 05-Feb-2019].

[19] M. Bor and U. Roedig, "LoRa transmission parameter selection," *Proc. - 2017 13th Int. Conf. Distrib. Comput. Sens. Syst. DCOSS 2017*, vol. 2018-Janua, pp. 27–34, 2018, doi:10.1109/DCOSS.2017.10, ISBN: 9781538639917, ISSN:2325-2944.

[20] M. Slabicki, G. Premsankar, and M. Di Francesco, "Adaptive configuration of lora networks for dense IoT deployments," *IEEE/IFIP Netw. Oper. Manag. Symp. Cogn. Manag. a Cyber World, NOMS 2018*, pp. 1–9, 2018, doi:10.1109/NOMS.2018.8406255, ISBN: 9781538634165.

[21] J. R. B. Junior, J. Lau, L. De Oliveira Rech, A. S. Morales, and R. Moraes, "Experimental Evaluation of the Coexistence of IEEE 802.11 EDCA and DCF Mechanisms," *Proc. - IEEE Symp. Comput. Commun.*, vol. 2018-June, pp. 847–852, 2018, doi:10.1109/ISCC.2018.8538640, ISBN: 9781538669501, ISSN:15301346.

[22] C. Pham, "Investigating and experimenting CSMA channel access mechanisms for LoRa IoT networks," in *IEEE Wireless Communications and Networking Conference, WCNC*, 2018, vol. 2018-April, pp. 1–6, doi:10.1109/WCNC.2018.8376997, ISBN:9781538617342, ISSN: 15253511.

[23] P. Yuan, X. Wen, H. Lu, and Q. Pan, "Dynamic Backoff Based Access Mechanism for LoRaWAN Class A," in *IEEE International Conference on Energy Internet Dynamic*, 2018, pp. 219–223, doi:10.1109/ICEI.2018.00047, ISBN:9781538641316, ISSN: 15502368.

[24] Y. Song, O. Zendra, and O. Zendra, "Using Cooja for WSN Simulations : Some New Uses and Limits To cite this version : Using Cooja for WSN Simulations : Some New Uses and Limits," pp. 319–324, 2016, ISBN: 9780994988607.

[25] O. Georgiou and U. Raza, "Low Power Wide Area Network Analysis: Can LoRa Scale?," *IEEE Wirel. Commun. Lett.*, vol. 6, no. 2, pp. 162–165, 2017, doi:10.1109/LWC.2016.2647247, ISSN:21622345.

[26] "Discrete Event Network Simulator - NS3." [Online]. Available: https://www.nsnam.org/. [Accessed: 08-May-2019].

[27] "NetSim - Network Simulator &amp; Emulator." [Online]. Available: https://www.tetcos.com/download.html. [Accessed: 08-May-2019].

[28] "SimPy 3.0.11." [Online]. Available: https://simpy.readthedocs.io/en/latest/. [Accessed: 08-May-2019].

[29] "Opnet.com." [Online]. Available: http://www.opnet.com/. [Accessed: 08-May-2019].

[30] "Crypto++ Library 8.0." [Online]. Available: https://www.cryptopp.com/. [Accessed: 05-Feb-2019].

[31] "Eclipse." [Online]. Available: https://www.eclipse.org/. [Accessed: 08-May-2019].

[32] R. Sanchez-Iborra, J. Sanchez-Gomez, J. Ballesta-Viñas, M. D. Cano, and A. F. Skarmeta, "Performance evaluation of lora considering scenario conditions," *Sensors (Switzerland)*, vol. 18, no. 3, 2018, doi:10.3390/s18030772, ISSN:14248220.

[33] J. D. Parsons, *The Mobile Radio Propagation Channel*. 1992, doi:10.1111/1365-2435.13050ISBN:047198857X.

[34] J.-M. Martinez-Caro and M.-D. Cano, "A holistic approach to evaluate the performance of applications and services in the Internet of Things," *Submitt. to Int. J. Commun. Syst.*, no. Special issue on: Emerging ICT Applications and Service-Big Data, IoT, and Cloud Computing, 2019.

[35] "Euskadi air quality (2018)," 2018. [Online]. Available: http://opendata.euskadi.eus/catalogo/-/calidad-aire-en-euskadi-2018/. [Accessed: 09-Sep-2019].

[36] Q. Wu *et al.*, "Cognitive internet of things: A new paradigm beyond connection," *IEEE Internet Things J.*, vol. 1, no. 2, pp. 129–143, 2014, doi:10.1109/JIOT.2014.2311513, ISBN: 2327-4662 VO - 1, ISSN:23274662.