

# LR-FHSS-Sim: A Discrete-Event Simulator for LR-FHSS Networks

Jean Michel de Souza Sant'Ana, Arliones Hoeller Jr., Hirley Alves, Richard Demo Souza

*Centre for Wireless Communications, University of Oulu, Oulu, Finland*

*Telecommunications Engineering Department, Federal Institute of Santa Catarina, São José, Brazil*

*Electrical & Electronics Engineering Department, Federal University of Santa Catarina, Florianópolis, Brazil*

{jean.desouzasantana, hirley.alves}@oulu.fi, arliones.hoeller@ifsc.edu.br, richard.demo@ufsc.br

**Abstract**—This work presents the LR-FHSS-Sim, a free and open-source discrete-event simulator for LR-FHSS networks. We highlight the importance of network modeling for IoT coverage, especially when it is needed to capture dynamic network behaviors. Written in Python, we present the LR-FHSS-Sim main structure, procedures, and extensions. We discuss the importance of a modular code, which facilitates the creation of algorithmic strategies and signal-processing techniques for LR-FHSS networks. Moreover, we showcase how to achieve results when considering different packet generation traffic patterns and with a previously published extension. Finally, we discuss our thoughts on future implementations and what can be achieved with them.

**Index Terms**—Internet-of-Things, LR-FHSS, discrete-event simulation, open-source, traffic generation.

## I. INTRODUCTION

Non-terrestrial networks (NTN) can enable global access to the Internet of Things (IoT) services regardless of geographical location or terrain. Services in remote areas like transportation, fleet management, logistics, solar, oil and gas extraction, offshore monitoring, utilities smart metering, farming, environment monitoring, and mining are some examples [1]. Using Low Power Wide Area Networks (LPWANs) combined with satellite technologies is a promising solution to integrate terrestrial networks with NTN [2]. A promising technology for satellite communication for IoT applications is the Long Range Frequency Hopping Spread Spectrum (LR-FHSS) [3], [4]. As part of the Long Range Wide Area Network (LoRaWAN) specification [5] but focused on satellite communication, this technique divides the payload into small pieces and transmits it across different physical channels. On top of that, it also transmits several redundant copies of the header into different frequency bandwidth channels.

Research on LR-FHSS shows us that modeling such networks can be quite challenging, where one can make several assumptions to make a simpler model [6], or work with very detailed models [7] even considering the channel effects due to satellite movement. Additionally, the adoption of “smart” signal processing and resource allocation techniques is increasing as 6G research moves toward AI integration [8]. All of this can be impractical to describe with relatively simple mathematical frameworks [9]. The use of simulations is quite common, mostly based on discrete Monte Carlo techniques, usually tailor-made to the problem being studied [2], [4]. This

raises some issues like low re-usability and low flexibility for future research.

An alternative approach is the use of discrete-event simulations. The main idea of this technique is to model a system whose global state changes over time. This is relevant, for example, in cases where there is a strong temporal correlation (e.g., sequential transmissions, different transmission durations), especially because they are not easily modeled by relatively simple mathematical frameworks or Monte Carlo simulations. Moreover, in discrete-event simulations, isolating and analyzing parts of a system and changing specific network functionalities without compromising or even changing the rest of the setup becomes easier. Frameworks like ns-3 [10] and OMNeT++ [11], for example, implement robust discrete-event simulations for several different technologies and environments, from traditional TCP/UDP internet to complex wireless network environments. Alternatively, Python presents a package called SimPy [12], which allows the creation of customized simulation environments.

The development and use of discrete-event simulators for LPWAN, particularly for the Chirp Spread Spectrum (CSS) LoRa modulation [13], is very common in the literature. We can cite at least four different modules developed for ns-3 [14]–[22]<sup>1 2 3</sup>, the FLoRa (Framework for LoRa) for the OMNeT++ [23]–[27]<sup>4</sup> and the LoRaSim [28], [29]<sup>5</sup> built with the SimPy framework for Python. From the listed simulators, FLoRa and the LoRaWAN ns-3 module [14] are the most organized and structured, from which we may attribute their success. However, the modules and simulation setups with them are not centralized, while the original repository still contains only the basic modules. Although LoRaSim was not officially published on a repository and its last update was in 2018, it was the most used and extended LoRa simulator [30]–[40]. We can refer to its popularity partially due to being the first simulator openly available for LoRa networks. However, even more recent works still considered using it, highlighting that its simplicity may be a decisive factor in such a choice.

<sup>1</sup><https://github.com/signetlabdei/lorawan>

<sup>2</sup><https://github.com/drakkar-lig/lorans3-module>

<sup>3</sup><https://github.com/imec-idlab/ns-3-dev-git/tree/lorawan>

<sup>4</sup><https://flora.aalto.fi>

<sup>5</sup><https://mcbor.github.io/lorasim/>

TABLE I  
LIST OF REPRESENTATIVE LoRA PUBLICATIONS USING DISCRETE-EVENT  
SIMULATORS GROUPED IN DIFFERENT CATEGORIES.

Category	Publications
Performance and scalability analysis	[14], [16], [21], [26], [28], [30], [32], [33], [38]
Access and scheduling algorithms	[17], [18], [19], [20], [24], [34], [35], [36], [39]
Parameters optimization	[15], [22], [23], [25], [27], [37], [40]
Multiple gateways analysis	[29], [31]

We list in Table I what we consider to be the main categories of works utilizing discrete-event simulators for LoRa networks, along with a list of some representative papers in each category. As expected, several works on performance and scalability analysis are used since they tend to be the first step when validating a simulator. However, we can see many works on scheduling algorithms, optimizations, and even multiple gateways. These techniques can be very challenging to model only mathematically. For example, scheduling and online parameter optimization algorithms tend to evolve with time until they reach a certain stability. Ultimately, in the case of dynamic networks, mathematical modeling might become of utmost complexity. Finally, the analysis of multiple-gateway scenarios is very dependent on the position of the gateways; thus, closed-form equations are impractical in most cases.

#### A. Contributions

Drawing inspiration from LoRaSim for CSS LoRa, this work introduces an open-source LR-FHSS discrete-event simulator coded in Python utilizing the SimPy framework. Unlike prior LR-FHSS works, our simulation environment is not tailored to specific scenarios and can accommodate various algorithms and signal processing techniques. Unlike LoRaSim, we have enhanced the code structure and publication methods to foster usability across diverse research projects. Unlike FLoRa, LR-FHSS-Sim is designed to be modular, allowing researchers to employ the necessary structures for simulations selectively and develop new modules with additional functionalities.

#### B. Organization

This paper is organized as follows. In Section II, we present a brief description of the LR-FHSS technology, the simulator architecture, and an example of an extension module. Section III presents some results with the simulator for a baseline model, reproducing one published result using one extension and an example of a novel extension module. Finally, Section IV concludes the paper.

## II. LR-FHSS-SIM

This section presents the LR-FHSS-Sim (LoRa - Frequency Hopping Spread Spectrum Simulator). The simulator is avail-

able freely on its online repository<sup>6</sup>. However, before diving into it, we discuss some basic concepts of LR-FHSS.

LR-FHSS is a Frequency-Hopping Spread Spectrum technique, part of the LoRaWAN specification, designed mainly for uplink IoT direct-to-satellite communications [41]. An LR-FHSS transmission hops over multiple frequencies (the so-called physical channels) while employing GMSK modulation at a bandwidth of 488 Hz. The total bandwidth of the LR-FHSS channel and the minimum physical channel separation of the LR-FHSS hopping depend on regional specifications [5]. This results in a different number of physical channels of 488 Hz available for use during one single transmission. For example, in Europe, there are 8 options of 35 or 86 physical channels to hop, depending on the chosen data rate. These options are the so-called channels' grids, illustrated in [3], [9].

The LR-FHSS packet structure comprises a header and a payload [5]. The header contains metadata about the transmission parameters and, most importantly, the seed to generate the pseudorandom hop sequence defining the physical channels assigned to the packet elements during that transmission. Usually, the end device sends two or three header copies of  $t_{header} = 233.472$  ms sequentially across different physical channels. The payload is channel coded according to a convolutional code of coding rate (CR) of 1/3 or 2/3 and split into small fragments of duration  $t_f = 102.4$  ms [5]. The number of payload fragments is given by [9]

$$f = \left\lceil \frac{b+3}{6 \text{ CR}} \right\rceil, \quad (1)$$

where  $b$  is the payload length (in bytes) and  $\text{CR} \in \{1/3, 2/3\}$  represents the coding rate.

Data transmission follows the unslotted ALOHA multiple access protocol, where end devices randomly pick one seed and generate a pseudorandom sequence of physical channels. During transmission, the end device hops to a new physical channel for every packet element (headers and fragments). A receiver that successfully decodes a header can recover the pseudorandom sequence and acquire knowledge of the physical channels used for the transmission. Finally, a packet is successfully decoded if at least one header and 1/3 or 2/3 of the total fragments are received, depending on the employed coding rate.

#### A. Architecture

The LR-FHSS-Sim mainly comprises some core classes in the file `lrfhss_core.py`. These classes represent the main structures of the simulator. They are depicted in Fig. 1 and described below:

- **Fragment:** The Fragment class is the simplest structure in the simulator. It contains its own information about duration, the Packet to which it belongs, its type (header or payload), and if it was transmitted, received with

<sup>6</sup><https://github.com/Xexell/LR-FHSS-sim>

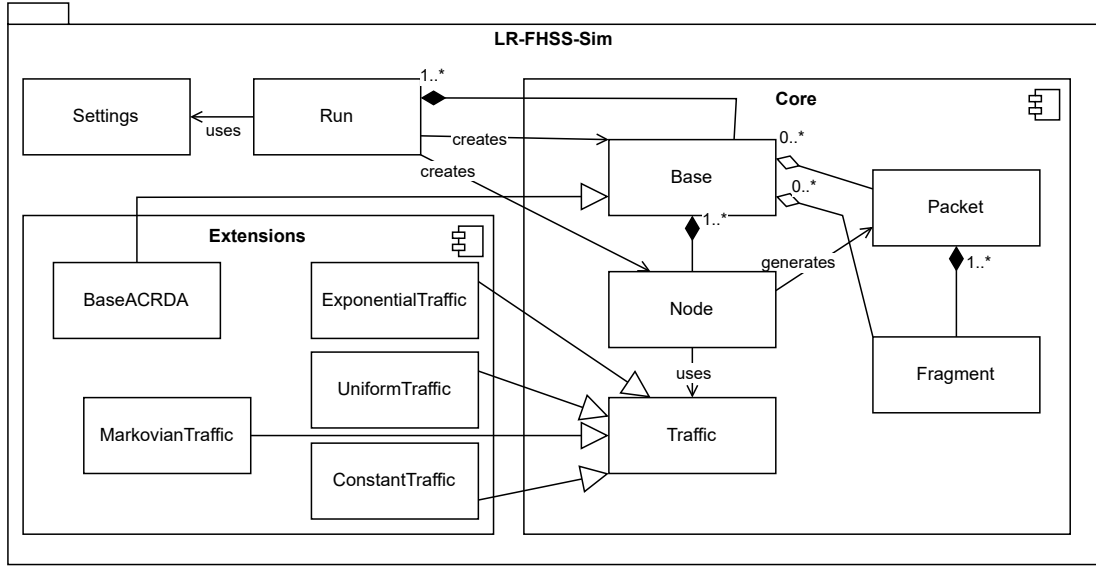


Fig. 1. Simplified UML [42] diagram of the LR-FHSS-Sim package, showing the Core and Extensions components and their respective classes.

success, and, in case of collision, the information about the other collided fragments.

- **Packet:** The Packet class generates the packet structure with headers and payload fragments using the Fragment class. It also selects the OBW channels for each of its fragments and can deliver the next Fragment to be transmitted. The packet also contains information about the Node to which it belongs and if it was successfully received.
- **Node:** The Node class represents the end devices and is the one with the transmission routine. This routine runs until the end of the simulation time for each end device in the network. The transmission routine triggers all other parts of the simulation. The Node generates one new Packet to be transmitted following a specific traffic pattern, implemented by the Traffic class. Finally, it has information on the number of Packets transmitted.
- **Traffic:** Traffic is an abstract class used by the Node class and responsible for generating the packet inter-arrival intervals for each node following a specific traffic model. Since it is common in network simulations to evaluate different traffic models, we choose to design the traffic model as an extensible component. The traffic model class includes functions that return the interval, in seconds, to the next packet transmission. Different traffic models have been developed as extensions, as discussed later in the paper.
- **Base:** The Base class represents the receiver, i.e., the gateway. It contains information on the transmitted fragments and evaluates whether a fragment transmission was successful. During the configuration of the simulation, instances of the Node class are aggregated to the Base, so they can be processed during simulation. In the case of the simple Base class, it checks if there is a collision,

considering that all end devices are in the Base coverage area. Moreover, it tries to decode packets when their transmission ends according to the coding rate used and stores information about how many successful packet transmissions each node made.

There are two other files in the repository for running a simple simulation. Although they are not exactly parts of the simulation package, we provided them as examples in case someone needs a starting point when building their simulations. The first file is `settings.py`. Here, inspired by the Parametric Singleton Design Pattern [43], we defined a Settings class that has all the necessary parameters to run a simulation. It makes simulations easy, as we calculate the number of headers and fragments automatically, and it presents standard inputs for all simulation parameters. Thus, one can focus only on the variables in which they are interested.

The later file is `run.py`, where we design the simulation using the network components and possibly return the numeric results we seek.

To generate the simplest network, one needs to create the SimPy environment, the Base class, and the Nodes (end devices), add each node to the Base, add the Node transmit method to the environment process handler, and finally start the simulation for a determined amount of time. The results can be obtained by accessing the network objects of interest. All these steps and others, such as selecting different extensions, are present in the `run.py` file.

### B. Extensions

An interesting feature of the simulator is the development of extensions of different signal processing and machine learning algorithms and aspects of network modeling, like traffic patterns, channel modeling, and others. Here, we present two extensions to serve as examples for future developments.

1) *Traffic Modeling*: The first set of extensions refers to the transmission generation model, stored in the file `traffic.py`. Note that the traffic generator is a mandatory part of the simulation, so at least one version of this class must be available. The selection of the traffic model is made by adjusting the corresponding parameter in the `settings.py` file. For example, one implementation models the inter-arrival time as an exponentially distributed random variable, corresponding to a Poisson traffic model, possibly the most used traffic model in simulations and mathematical modeling. We have also modeled the inter-arrival time as a uniform random variable and a constant value with a small Gaussian drift to avoid collisions by the same end devices.

These three options model sparse and uniform traffic in the long term. To model less uniform traffic, we implemented a simple two-state Markov chain traffic model inspired by [44]. We consider State 0 as not transmitting and State 1 as transmitting. The probability of staying in State 0 is  $p_i$ , and the probability of going from State 1 to State 0 is  $q_i$ . Since this is a discrete model, we have to consider that each step represents a certain amount of time  $S_M = T\pi_1$ , where  $T$  is the desired average inter-arrival time and  $\pi_1$  is the State 1 steady-state probability given as

$$\pi_1 = \frac{p_i - 1}{p_i - q_i - 1}. \quad (2)$$

Note that small values of  $p_i$  imply in many transmissions (or small  $T$  for a fixed  $S_M$ ), and values of  $q_i \ll p_i$  represent traffic with burst behavior, where there is a higher probability of transmitting several messages subsequently.

2) *Asynchronous Contention Resolution Diversity Aloha (ACRDA)*: A second extension, is the recently proposed ACRDA-enabled LR-FHSS network [9], contained in file `acrda.py`. For this extension, we need to implement a receiver with ACRDA capabilities. To do so, we extended the Base class into the BaseACRDA class and modified it accordingly. In short, we override the core Base decoding method by implementing the ACRDA decoding procedure. Moreover, we included in the BaseACRDA a memory buffer for packets and a window procedure to be added as an independent process in the SimPy environment, similar to the Node transmission. This configuration is in the `run.py` file.

### III. RESULTS

In this section, we present some numerical results acquired using the LR-FHSS-Sim. We start with some basic results, then compare the traffic modules presented in the previous section and show the impact when using different models. Finally, we compare different traffic models when using the ACRDA-LR-FHSS extension. For all results, whenever not explicitly mentioned, we considered DR8 (3 headers and CR 1/3), 20 bytes of payload, simulation time of 24 hours, 100 iterations, average interval between transmissions  $T$  of 900 s (4 transmissions per hour), 1 grid of 35 OBW channels, where we scale the total number of end devices multiplying by 8 (considering 8 different channel grid available), ACRDA

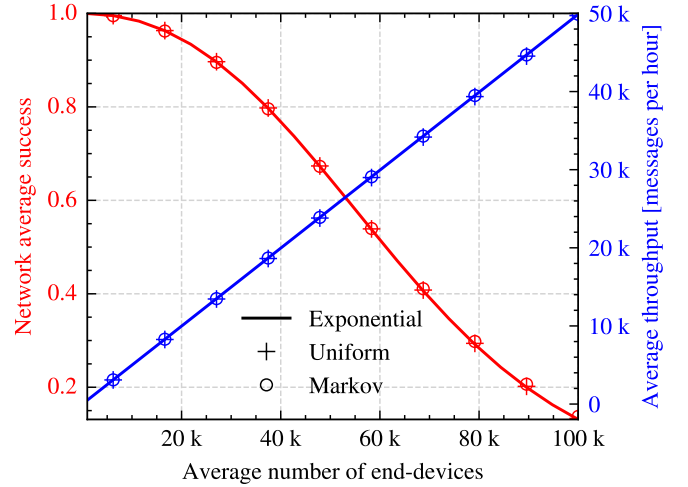


Fig. 2. Network average success (red left) and throughput (blue right) for regular LR-FHSS network for different number of end devices  $N$  with different traffic models.

normalized window size of 2 and normalized window step of 0.5. For the two-state Markov traffic model we used  $p_i = 0.99998$  and  $q_i = 0.15$ . These two last parameters, used as an example, present a network with a burst traffic behavior, with a high probability of transmitting messages in sequence. It is important to note that we validate the simulator with literature theoretical results in [9], where the network average success behavior follows a basic mathematical framework.

In Fig. 2, we present the network's average success rate and throughput when using different device traffic models for different numbers of end devices. In blue, it is important to see that the average network throughput remains the same for all models, which means that the simulator input value of the average interval between transmissions holds true for all models. Moreover, in red, we can see that the different traffic models almost did not impact the average success of the standard LR-FHSS network. This makes sense since we are considering a long-run simulation of 24 hours with a large number of end devices. Thus, it has enough time to generate the average traffic result.

Fig. 3 depicts the cumulative distribution of the end device success probability for different traffic models when using a 5-hour simulation run. We used a reduced simulation run window to better visualize the variance caused by the Markov traffic. Here, we can see that, despite the network average success of both traffic models being the same, they present some differences. The Markov traffic model presents a slightly higher variance in the end device probability than the exponential model. This is due to the Markov traffic presenting a more unpredictable traffic generation with burst behavior. This shows us that even though the average result is similar, we still impact the performance of the end devices, as there will be more devices with lower reliability instead of a more uniform distribution. This result is relevant because we consider a case where all devices are in coverage; thus, they present the same

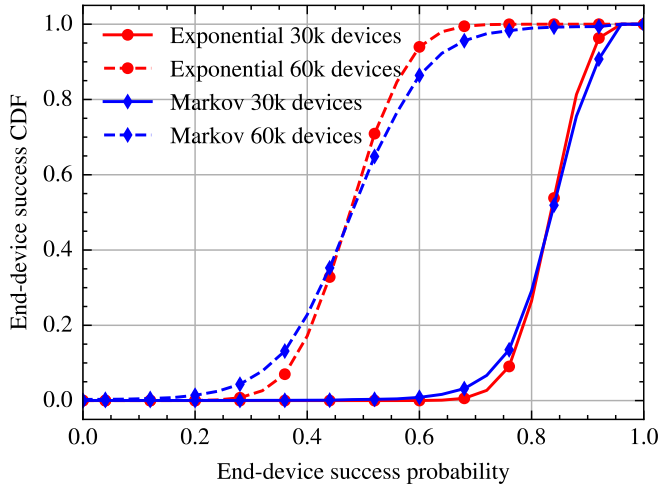


Fig. 3. Cumulative distribution function of the end device success probability for different traffic models and number of end devices with 5 hours of simulation time.

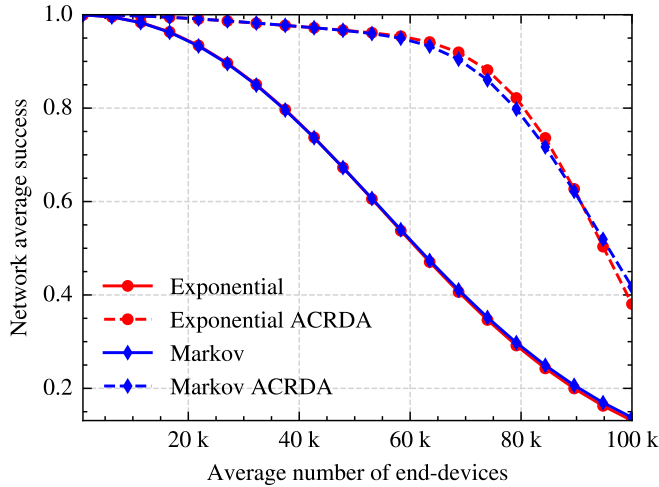


Fig. 4. Network average success for regular LR-FHSS network and ACRDA-based LR-FHSS [9] for different number of end devices  $N$  with different traffic models.

behavior, where the traffic distribution is the only difference.

Finally, Fig. 4 presents the network average success of the ACRDA module [9] when using the exponential and two-state Markov traffic modules. Unlike Fig. 2, we can see a difference in network average success between the two traffics when using the ACRDA. This result comes from the fact that devices usually tend to transmit many messages in a sequence when using Markov traffic with burst behavior. Thus, whenever two or more devices transmit simultaneously, they are expected to transmit new messages and more collisions occur. All these transmitted messages subsequently impose difficulties on the ACRDA mechanism, especially with a limited decoding sliding window, when applying the interference cancellation method and recovering the collided messages.

## IV. CONCLUSIONS

In this work, using the SimPy framework, we introduced the LR-FHSS-Sim, an open-source discrete-event simulator for LR-FHSS networks in Python. We presented the simulator architecture, how it works, and how to develop new functionalities. Moreover, we presented a traffic model extension, where we can simulate different traffic patterns for the end devices and evaluate the results regarding average network success and network throughput. We showed that traffic models can present similar average performance regarding success and throughput, but there may be relevant differences in other performance statistics. We hope that LR-FHSS-Sim can become a useful tool for the wireless community and be further developed to include other practical extensions.

## ACKNOWLEDGEMENTS

This work or its authors have been partially supported in Finland by the Research Council of Finland (former Academy of Finland) 6G Flagship Programme (Grant Number: 346208), and by the European Union through the Interreg Aurora project ENSURE-6G (Grant Number: 20361812).; and in Brazil by CNPq (Grant Numbers: 402378/2021-0 and 305021/2021-4) and RNP/MCTIC 6G Mobile Communications Systems (Grant Number: 01245.010604/2020-14).

## REFERENCES

- [1] M. Centenaro, C. E. Costa, F. Granelli, C. Sacchi, and L. Vangelista, "A survey on technologies, standards and open challenges in satellite IoT," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1693–1720, 2021.
- [2] M. Asad Ullah, K. Mikhaylov, and H. Alves, "Massive machine-type communication and satellite integration for remote areas," *IEEE Wireless Communications*, vol. 28, no. 4, pp. 74–80, 2021.
- [3] J. A. Fraire, A. Guitton, and O. Iova, "Recovering headerless frames in LR-FHSS," Jun. 2023, arXiv:2306.08360 [cs]. [Online]. Available: <http://arxiv.org/abs/2306.08360>
- [4] G. Alvarez, J. A. Fraire, K. A. Hassan, S. Céspedes, and D. Pesch, "Uplink transmission policies for LoRa-based direct-to-satellite IoT," *IEEE Access*, vol. 10, pp. 72 687–72 701, 2022.
- [5] *LoRaWAN Regional Parameters*, LoRa Alliance Technical Committee Regional Parameters Workgroup, Sep. 2022, rP002-1.0.4. [Online]. Available: <https://resources.lora-alliance.org/technical-specifications/rp002-1-0-4-regional-parameters>
- [6] M. A. Ullah, K. Mikhaylov, and H. Alves, "Analysis and simulation of LoRaWAN LR-FHSS for direct-to-satellite scenario," *IEEE Wireless Communications Letters*, vol. 11, no. 3, pp. 548–552, 2022.
- [7] A. Maleki, H. H. Nguyen, and R. Barton, "Outage probability analysis of LR-FHSS in satellite IoT networks," *IEEE Communications Letters*, vol. 27, no. 3, pp. 946–950, 2023.
- [8] K. B. Letaief, Y. Shi, J. Lu, and J. Lu, "Edge artificial intelligence for 6G: Vision, enabling technologies, and applications," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 1, pp. 5–36, 2022.
- [9] J. M. S. Sant'Ana, O. S. Neto, A. Hoeller, J. L. Rebelatto, R. D. Souza, and H. Alves, "Asynchronous contention resolution-aided ALOHA in LR-FHSS networks," *IEEE Internet of Things Journal*, pp. 1–1, 2024.
- [10] ns-3 network simulator. [Online]. Available: <https://www.nsnam.org/>
- [11] OMNeT++ discrete event simulator. [Online]. Available: <https://omnetpp.org/>
- [12] SimPy discrete event simulation for python. [Online]. Available: <https://simpy.readthedocs.io/>
- [13] *LoRaWAN™ L2 1.0.4 Specification*, LoRa Alliance, Inc., Oct. 2020. [Online]. Available: <https://resources.lora-alliance.org/technical-specifications/ts001-1-0-4-lorawan-l2-1-0-4-specification>
- [14] D. Magrin, M. Centenaro, and L. Vangelista, "Performance evaluation of LoRa networks in a smart city scenario," in *2017 IEEE International Conference on Communications (ICC)*, 2017, pp. 1–7.

- [15] 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)*, 2017, pp. 1–6.
- [16] 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*, vol. 4, no. 6, pp. 2186–2198, 2017.
- [17] 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.
- [18] T.-H. To and A. Duda, "Simulation of LoRa in ns-3: Improving LoRa performance with CSMA," in *2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1–7.
- [19] M. Capuzzo, D. Magrin, and A. Zanella, "Confirmed traffic in LoRaWAN: Pitfalls and countermeasures," in *2018 17th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, 2018, pp. 1–7.
- [20] J. Haxhibeqiri, I. Moerman, and J. Hoebeke, "Low overhead scheduling of LoRa transmissions for improved scalability," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3097–3109, 2019.
- [21] D. Magrin, M. Capuzzo, and A. Zanella, "A thorough study of LoRaWAN performance under different parameter settings," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 116–127, 2020.
- [22] J. Finnegan, R. Farrell, and S. Brown, "Analysis and enhancement of the LoRaWAN adaptive data rate scheme," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7171–7180, 2020.
- [23] M. Slabicki, G. Premasankar, and M. Di Francesco, "Adaptive configuration of LoRa networks for dense IoT deployments," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, 2018, pp. 1–9.
- [24] L. Leonardi, F. Battaglia, and L. Lo Bello, "RT-LoRa: A medium access strategy to support real-time flows over LoRa-based networks for industrial IoT applications," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10812–10823, 2019.
- [25] G. Premasankar, B. Ghaddar, M. Slabicki, and M. D. Francesco, "Optimal configuration of LoRa networks in smart cities," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 12, pp. 7243–7254, 2020.
- [26] V. Toro-Betancur, G. Premasankar, M. Slabicki, and M. Di Francesco, "Modeling communication reliability in LoRa networks with device-level accuracy," in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, 2021, pp. 1–10.
- [27] G. G. M. de Jesus, R. D. Souza, C. Montez, and A. Hoeller, "LoRaWAN adaptive data rate with flexible link margin," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 6053–6061, 2021.
- [28] M. Bor, U. Roedig, T. Voigt, and J. Alonso, "Do LoRa low-power wide-area networks scale?" in *MSWiM '16 Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. ACM Press, Nov. 2016, pp. 59–67.
- [29] T. Voigt, M. Bor, U. Roedig, and J. Alonso, "Mitigating inter-network interference in LoRa networks," in *EWSN '17 Proceedings of the 2017 International Conference on Embedded Wireless Systems and Networks*. ACM Press, Feb. 2017, pp. 323–328.
- [30] A.-I. Pop, U. Raza, P. Kulkarni, and M. Sooriyabandara, "Does bidirectional traffic do more harm than good in LoRaWAN based LPWA networks?" in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, 2017, pp. 1–6.
- [31] M. O. Farooq and D. Pesch, "Evaluation of multi-gateway LoRaWAN with different data traffic models," in *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*, 2018, pp. 279–282.
- [32] S. Sugianto, A. A. Anhar, R. Harwahu, and R. F. Sari, "Simulation of mobile lora gateway for smart electricity meter," in *2018 5th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, 2018, pp. 292–297.
- [33] C. M. S. Ferreira, R. A. R. Oliveira, and J. S. Silva, "Low-energy smart cities network with LoRa and bluetooth," in *2019 7th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, 2019, pp. 24–29.
- [34] G. Lee and J. Youn, "Group-based transmission scheduling scheme for building LoRa-based massive IoT," in *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIIC)*, 2020, pp. 583–586.
- [35] S. Cui and I. Joe, "Collision prediction for a low power wide area network using deep learning methods," *Journal of Communications and Networks*, vol. 22, no. 3, pp. 205–214, 2020.
- [36] Y. Lalle, M. Fourati, L. C. Fourati, and J. P. Barraca, "Routing strategies for LoRaWAN multi-hop networks: A survey and an SDN-based solution for smart water grid," *IEEE Access*, vol. 9, pp. 168 624–168 647, 2021.
- [37] L. Charles, B. Isong, F. Lugayizi, and A. M. Abu-Mahfouz, "Empirical analysis of LoRaWAN-based adaptive data rate algorithms," in *IECON 2021 - 47th Annual Conference of the IEEE Industrial Electronics Society*, 2021, pp. 1–7.
- [38] S. Francisco, P. Pinho, and M. Luís, "Improving LoRa network simulator for a more realistic approach on LoRaWAN," in *2021 Telecoms Conference (ConfTELE)*, 2021, pp. 1–6.
- [39] K. Wongwatthanaroek and R. Silapunt, "Transmission sequencing to improve LoRaWAN performance," in *2021 18th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, 2021, pp. 1–5.
- [40] H. Wang, S. Lv, Y. Han, X. Zhang, Y. Zhang, W. Dong, J. Liao, and H. Luan, "An energy-saving LoRa linear network system with adaptive transmission parameter," *IEEE Open Journal of the Industrial Electronics Society*, vol. 4, pp. 476–485, 2023.
- [41] G. Boquet, P. Tuset-Peiró, F. Adelantado, T. Watteyne, and X. Vilajosana, "LR-FHSS: Overview and Performance Analysis," *IEEE Communications Magazine*, vol. 59, no. 3, pp. 30–36, 2021.
- [42] M. Fowler, *UML Distilled: A Brief Guide to the Standard Object Modeling Language*, 3rd ed. Addison-Wesley Professional, 2003.
- [43] D. Lyon and F. Castellanos, "The parametric singleton design pattern," *Journal of Object Technology*, vol. 6, pp. 13–23, 03 2007.
- [44] F. Qasmi, M. Shehab, H. Alves, and M. Latva-aho, "Fixed rate statistical qos provisioning for markovian sources in machine type communication," in *2019 16th International Symposium on Wireless Communication Systems (ISWCS)*, 2019, pp. 474–479.