ELoRa: End-to-end Emulation of Massive IoT LoRaWAN Infrastructures

Alessandro Aimi, Stephane Rovedakis, Fabrice Guillemin, Stefano Secci

ELoRa: End-to-end Emulation of Massive IoT LoRaWAN Infrastructures

Alessandro Aimi*†, Stéphane Rovedakis†, Fabrice Guillemin*, and Stefano Secci†
*Orange Innovation, France. {alessandro.aimi, fabrice.guillemin}@orange.com
†Cedric, Cnam, France. {alessandro.aimi, stephane.rovedakis, stefano.secci}@cnam.fr

***Abstract*—In this paper, we present *ELoRa*, an emulation tool that generates Long Range Wide Area Network (LoRaWAN) traffic for an arbitrary number of LoRa devices and in an end-to-end virtualized LoRaWAN setting. Using ns-3, we improve an existing radio access network simulator to produce traffic compatible with ChirpStack, an open-source, cloud-native, LoRaWAN network functions stack. Our tool can be used to create realistic traffic and anomalies in order to test orchestration techniques on a real, distributed infrastructure. Moreover, the LoRaWAN core network functions (bridges, network server) are agnostic to the simulation of the radio access and can change parameters of simulated devices using native LoRaWAN protocol primitives, therefore enabling live-testing of resource allocation techniques to manage the radio access network. Multiple *ELoRa* instances can be connected to the same LoRaWAN core, each instance being able to support 50000 devices and 7 gateways.**

***Index Terms*—LoRaWAN, Massive IoT, E2E, Emulation**

I. Introduction

In the growing Internet of Things (IoT), Long Range Wide Area Networks (LoRaWANs) [1] has become a popular solution for environment sensing and monitoring thanks to its cheap and easy-to-operate nature. Due to the unavailability of large-scale real-life testbeds, many simulators have been developed to study LoRaWAN at scale [2], [3]. These tools aim at accurately modeling the radio performance of devices and gateways, but do not cover the core network functions, in particular LoRaWAN network servers and bridging functions. On the other hand, only basic traffic generators currently exist for network servers [4] without, however, the possibility of producing non-trivial management scenarios involving the radio access network, resource allocations and traffic anomalies.

In this paper, we present *ELoRa* [5], a software tool aimed at accurately emulating end-to-end (E2E) LoRaWAN traffic, from device to server. *ELoRa* is built using the well known ns-3 network simulator [6], extending the LoRaWAN module presented in [2]. Among existing simulators, this module presents the most complete implementation of the LoRaWAN Medium Access Control (MAC) protocol. In our proposition, we build a translation layer between the simulation, and real (UDP-encapsulated) LoRaWAN traffic, *de facto* enabling two-way real-time communications with the outside environment. The resulting UDP traffic is transparently accepted by most LoRaWAN servers, as we re-implement in ns-3 the gateway packet forwarder protocol [7] developed by Semtech, which patented the Long Range (LoRa) modulation technology.

In our proposition, we focus on the well established ChirpStack open-source LoRaWAN network server [8]. To enable the server's full device management capabilities, we include a component exploiting the REST API of ChirpStack to seamlessly register devices on the server. In addition, we significantly improve the original simulator to produce traffic that is compatible with a real server. Changes include, but are not limited to, resolving multiple MAC layer inaccuracies, adding cryptographic capabilities to devices, as well as introducing a tool to emulate smart-city traffic as per [9], and implementing the generation of LoRaWAN `.pcap` packet captures that can be dissected in programs like Wireshark [10].

Thanks to the variety of modeling tools offered by ns-3, *ELoRa* makes it possible to test management techniques on LoRaWAN servers under countless realistic scenarios and loads. Furthermore, parameters of simulated devices can be controlled live by the server to test existing and novel radio resource management algorithms in an comprehensive E2E controlled environment. Thanks to the cloud-native architecture of Chirpstack, ELoRa can be deployed locally to the server or in a distributed fashion, standalone, in multiple instances, or accompanying other traffic flows from real networks.

The paper is organized as follows. The system architecture and contribution are illustrated in Section II. We detail an example scenario emulated with *ELoRa* in Section III. Concluding remarks are presented in Section IV.

II. System Architecture

ELoRa is a discrete-event simulation tool running in real-time and interacting with other processes in the operating system. It takes benefit of the typical ns-3 object-oriented C++ workflow, requiring the specification of all simulated elements in a main file that is then compiled and executed by command line. As usual for ns-3 simulations, it runs under GNU/Linux as a single-threaded process. The architecture of a Chirpstack deployment with *ELoRa* is shown in Figure 1. In the following, we detail the different elements, from left to right.

The simulated radio channel is defined in terms of path loss and delay, picking up one out of the numerous models offered by ns-3 [11]. Simulated IoT devices and gateways are placed in a three dimensional space. Position and mobility can also be set with several models, such as for instance, hexagonal tiling for gateways, and uniformly in range for devices [12]. In our implementation, devices can be set to send periodical or Poisson traffic, with different payload sizes. We include an

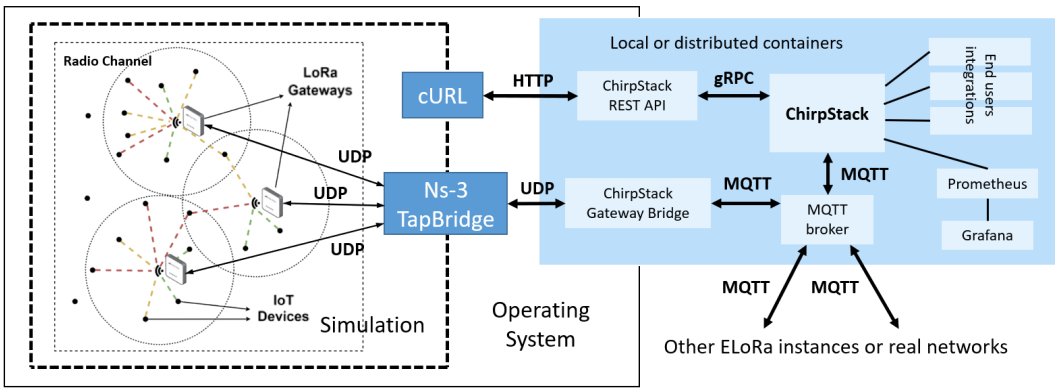

Fig. 1. Software architecture of an *ELoRa* deployment (*left*) on Chirpstack (*right*).

allocation helper for these parameters to emulate smart-city traffic as proposed in [9].

Currently the module supports Class A devices, the baseline LoRaWAN type, in the EU868 region. Devices comply with the frame format defined in the specifications [1], [13]. When compared with the original simulator, devices now implement all Class A mandatory MAC primitives. Using the cryptographic libraries from [14], we add message integrity code computation on uplink frames and the possibility to decrypt payloads from the server. Also, we introduce the option to export `.pcap` files of bidirectional traffic from the perspective of any device or gateway. In addition to the metrics produced by the ns-3 tracing system (e.g., packet delivery ratio, packet loss causes, channel utilization, energy consumption), such files can be used to examine live and in detail the serialized content of headers to show which MAC commands are exchanged.

All devices and gateways send frames through the same radio channel using multiple frequencies and modulation parameters. All interference computations happen according to the model in [2], with the added possibility of using the Signal to Interference Ratio (SIR) matrix from [15]. The different transmission parameters used are taken into account by the interference model. In the original simulator, interference between uplink and downlink frames is computed as if it was uplink on uplink. It has been shown that downlink transmissions retain an average 90% Packet Delivery Ratio (PDR) in the scenario of equal-power, concurrent uplink traffic [16]. For more accurate results, and in lack of a specific interference model, we consider downlink and uplink transmissions to be independent in terms of interference. Still, we maintain the assumption that gateways cannot receive uplink frames while busy transmitting downlink.

Uplink frames can be received by one or multiple gateways, which forward received traffic to core functions. The original module had a high level model of both the forwarding and the server. In our proposition, we implement in ns-3 the UDP packet forwarder protocol [7], integrating most libraries used in real gateways to communicate with servers. This lightweight application encapsulates LoRaWAN frames in UDP, and manages the synchronization of downlink transmissions from the server with reception windows of devices.

The UDP packets are sent to a final node using one of the connection models offered by ns-3, for instance 'IP over CSMA', and then they exit the simulation. This is made possible by the ns-3 `TapBridge` class on the final node. Ns-3 already serializes packets as if they were real ones, so the `TapBridge` node creates a tap interface in the underlying operative system and takes care of translating bidirectional traffic between the simulation and the operative system. Here, a ChirpStack Gateway Bridge can be deployed locally, or traffic can be forwarded to another machine hosting one. From this point, traffic enters the ChirpStack server infrastructure.

ChirpStack architecture is composed of different components that are generally deployed as containers. Components communicate with secure protocols (MQTT [17], gRPC [18]), and can be easily distributed to achieve Multi-access Edge Computing (MEC) goals. There can be multiple (MQTT) sources of LoRaWAN traffic, and gRPC is used to manage the server API. A separate translation component is given to provide a REST endpoint for the API. Most components expose metrics that can be exported with Prometheus [19] and observed in Grafana [20]. Finally, numerous end user integration schemes are available to exploit the data carried by frames or to elaborate additional metrics on the traffic.

To take full advantage of the server's network management features, i.e., to see metrics and enable parameter reconfiguration, devices and gateways need to be registered. For this reason, we develop an helper component using libcurl [21] to interact with the server's REST API. An API key can be generated directly from the server graphical interface and copied in the helper. All simulated objects are registered at the beginning of the simulation and then teared down on any interruption of the *ELoRa* process.

III. Example Scenario

We emulate 7 gateways positioned using hexagonal tiling at a distance of 5km from each other; 1000 static devices are placed uniformly at a maximal distance of 2.5km from any gateway, at an height between 1 and 10 meters. Their behaviour and payload are selected by following the distribution indicated in [9] for commercial devices in the city of New

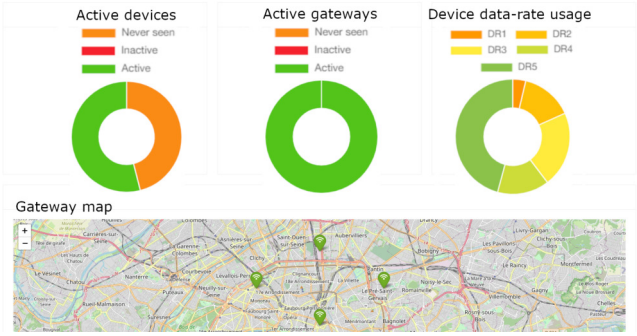

Fig. 2. The dashboard of ChirpStack.

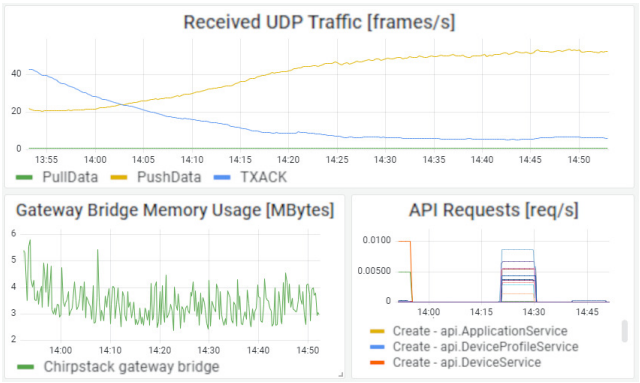

Fig. 3. Metrics monitoring example with Grafana.

York. Path loss is computed using the Okumura-Hata model for large urban areas with Rayleigh fading [12].

In Figure 2, we show the dashboard of ChirpStack when executing *ELoRa* with the described configuration. After 1 hour the server has detected more than an half of the devices, while devices with longer periodicity have only been registered. The server has increased the transmission data-rate of devices with the default Adaptive Data Rate (ADR) algorithm [3]. This helps reduce both interference and the impact of limitation on duty-cycle imposed on the EU868 band.

This is confirmed by Grafana metrics shown in Figure 3, where uplink UDP traffic increases, and the amount of acknowledgments to downlink MAC primitives decreases. The overall amount of traffic declines so the Chirpstack Gateway Bridge uses less memory. Finally, we can see the initial API calls of *ELoRa* to register simulated objects, and a second step representing internal API calls performed by the server.

In our testing, we were able to repeat this scenario with up to 50000 devices before being CPU bounded on an Intel Core i7-6600U @ 2.60GHz processor. During this limit case, the process was exhibiting a stable 950 MB of RAM usage.

IV. Conclusion

We have developed *ELoRa*, a tool to closely emulate LoRaWAN traffic that is capable of interacting with real server instances. *ELoRa* aims at helping research and industry actors to better understand the capabilities and limitations of LoRaWAN service infrastructures.

The flexibility of ns-3 enables emulation of real scenarios as well as the introduction of anomalies affecting the radio link and the server load. Radio parameters of simulated devices can be changed using the LoRaWAN protocol directly from the server API, allowing real-time experimentation with radio resource allocation algorithms. To our knowledge, *ELoRa* is the first tool for massive LoRaWAN simulation that presents these features, difficult to replicate with physical testbeds.

Consequently, orchestration platforms can be plugged in to test resource allocation techniques and improve the system's management automation capabilites under realistic loads. The overall framework can be easily scaled and distributed thanks to the cloud-native nature of ChirpStack.

Acknowledgment

This work was funded by the French ANR INTELLIGENTSIA project (grant nb: ANR-20-CE25-0011).

References

[1] *LoRaWAN L2 1.0.4 Specification*, TS001-1.0.4, LoRa Alliance, 2020.
[2] D. Magrin, M. Centenaro, and L. Vangelista, "Performance evaluation of LoRa networks in a smart city scenario," in *IEEE Int. Conf. Commun. (ICC)*, 2017, pp. 1–7.
[3] M. Slabicki, G. Premsankar, and M. Di Francesco, "Adaptive configuration of LoRa networks for dense IoT deployments," in *IEEE/IFIP Netw. Oper. Management Symp. (NOMS)*, 2018, pp. 1–9.
[4] *ChirpStack Simulator*. (2020). ChirpStack. Accessed: Jan. 13, 2023. [Online]. Available: https://github.com/brocaar/chirpstack-simulator
[5] *ELoRa*. (2023). Orange. Accessed: Mar. 2, 2023. [Online]. Available: https://github.com/non-det-alle/elora-docker
[6] *Ns-3 Network Simulator v3.37*. (2008). nsnam. Accessed: Jan. 13, 2023. [Online]. Available: https://www.nsnam.org
[7] *Lora network packet forwarder project v4.0.1*. (2017). Semtech. Accessed: Jan. 13, 2023. [Online]. Available: https://github.com/Lora-net/packet_forwarder
[8] *ChirpStack v4*. (2022). ChirpStack. [Online]. Accessed: Jan. 13, 2023. Available: https://www.chirpstack.io
[9] R. Huang, H. Li, B. Hamzeh, Y. Choi, S. Mohanty, and C. Hsu, "Proposal for evaluation methodology for 802.16p," IEEE 802.16 Broadband Wireless Access Work. Group, IEEE C802.16p-11/0102r2, May 2011.
[10] *Wireshark v4*. (2022). The Wireshark Foundation. Accessed: Jan. 13, 2023. [Online]. Available: https://www.wireshark.org
[11] M. Stoffers and G. Riley, "Comparing the ns-3 propagation models," in *20th IEEE Int. Symp. Modeling, Anal., Simul. Comput. Telecommun. Syst. (MASCOTS)*, 2012, pp. 61–67.
[12] A. Aimi, F. Guillemin, S. Rovedakis, and S. Secci , "Traffic control and channel assignment for quality differentiation in dense urban LoRaWANs," in *2022 20th Int. Symp. Modeling Optimization Mobile, Ad hoc, Wireless Netw. (WiOpt)*, 2022, pp. 153–160.
[13] *LoRaWAN Regional Parameters*, RP002-1.0.4, LoRa Alliance, 2022.
[14] *LoRaWAN end-device stack implementation and example projects v4.6*. (2015). Semtech. Accessed: Oct. 20, 2022. [Online]. Available: https://github.com/Lora-net/LoRaMac-node
[15] D. Croce, M. Gucciardo, S. Mangione, G. Santaromita, and I. Tinnirello, "Impact of LoRa imperfect orthogonality: Analysis of link-level performance," *IEEE Commun. Lett.*, vol. 22, no. 4, pp. 796–799, Apr. 2018.
[16] R. Saroui, A. Guitton, O. Iova, and F. Valois, "Uplink and downlink are not orthogonal in LoRaWAN!" in *IEEE 96th Veh. Technol. Conf. (VTC2022-Fall)*, 2022.
[17] *MQTT*. (2019). OASIS. Accessed: Jan. 13, 2023. [Online]. Available: https://mqtt.org
[18] *gRPC*. (2016). Google. Accessed: Jan. 13, 2023. [Online]. Available: https://grpc.io
[19] *Prometheus*. (2016). Cloud Native Computing Foundation. Accessed: Jan. 13, 2023. [Online]. Available: https://prometheus.io
[20] *Grafana*. (2014). Grafana Labs. Accessed: Jan. 13, 2023. [Online]. Available: https://grafana.com
[21] *libcurl*. (1996). curl. Accessed: Jan. 13, 2023. [Online]. Available: https://curl.se/libcurl