First Steps in the Development of a LoRaWAN Testbench

Federico Torres[1], Juan Soriano[1], and Guillermo Riva[1,2]

[1]Digital Communications Lab (LCD), National University of Cordoba
[2]GInTEA, National Technological University, Cordoba Regional Faculty
[1]insftorres@gmail.com, ju.soriano@gmail.com
[2]griva@frc.utn.edu.ar

***Abstract*—The development of Low Power Wide Area Networks (LPWANs) based on novel communication standards such as LoRaWAN has increased exponentially to provide network infrastructure to the large number of Internet of Things (IoT) solutions that are currently being developed. However, the first difficulty that developers face in the implementation and deployment of a LoRa network is the appropriate selection not only of the hardware but also the network server software. Likewise, it is essential to have a testbench to evaluate and configure the network optimally before its actual deployment. This is due to the large number of parameters that must be configured for proper operation of the network, which are dependent on the application scenario. In this paper, we will introduce the main features of LoRaWAN wireless communication technology, a comparison of the main features of hardware and network server software currently available. Finally, we show the first implementation and results of a LoRaWAN testbench and the difficulties detected.**

***Index Terms*—LPWAN, LoRaWAN, IoT, Testbench, LoRa Server**

I. Introduction

The development of LPWANs based on LoRaWAN communication protocol increased exponentially in last years to provide infrastructure to the large number of IoT solutions that are being designed. LoRaWAN networks complement technologies such as NarrowBand IoT (NB-IoT) or Long Term Evolution (LTE) Cat M1, mainly in areas far away from large cities, where there is no mobile communications infrastructure. Therefore, unlike the previous ones, LoRaWAN allows full control of the network since developers can access the information of each network element (node, gateway, network server).

The main difficulty that developers face when implementing a LoRaWAN network is not only the proper selection of *hardware* but also the *network server software* to be used. Likewise, it is essential to have a testbench to determine optimal configurations and network operation through the full observation of each network element before its actual deployment. There are several configuration parameters, such as, data rate, spreading factor, bandwidth, receive delays, join accept delays, etc. Efficiently monitoring a network requires full observability of the state of each element in the network. The main challenge lies in correlating the monitoring logs from all network components in a server and reconstructing useful network knowledge such as routing information, end-to-end performance, bottlenecks, etc. The idea of this testbench is deployed networks in vivo until confidence and experience has been acquired. By adjusting LoRaWAN parameters for a given deployment site iteratively, we can optimize network performance. The proposed testbench is also designed to be transportable to the field, allowing on-site testing.

Most of LoRaWAN's research works focus on understanding the limits of these networks by simulation or analytical approaches [1], [2]. Adelantado et al [3] provide an overview about the capabilities and limitations of LoRaWAN. Authors demonstrate that capacity and network size are limited by the duty-cycle. Therefore, open research challenges are described in this work. However, works related to monitoring frameworks that could be used for IoT networks such as LoRaWAN have been published recently.

$SensorLab2$ is a monitoring framework or in-the-field IoT validation platform for IoT networks in which nodes report events that are significant to the network life, following clear specified semantics [4]. Clear semantics allow maintaining a network model, run on a computer on the side of the network, mirroring the state of the real network. The successive states of this model are archived in a database and can be browsed by the investigation tools instead of querying the real physical network. This is especially beneficial in the case of low bandwidth networks or resource-constrained devices encountered in IoT. With two different examples, LoRaWAN and OpenWSN, authors illustrate how this monitoring framework allows using the same generic tool suite to debug, monitor and tweak a diversity of networks. To demonstrate the proposal, authors instrumented the OpenWSN network stack, from the Medium Access Control (MAC) layer (IEEE 802.15.4e) to the application layer (Constrained Application Protocol, CoAP) with emphasis on the IPv6 routing layer, IETF Routing Protocol for Low-Power and Lossy Networks (RPL). Similarly, authors have instrumented the reference LoRaWAN 1.0 stack to monitor a commercial LoRaWAN network deployed by the French telecommunications corporation Orange without customizing the monitoring framework.

The FIT (Future Internet of Things) IoT-LAB testbed [5] offers a first class facility with thousands of wireless nodes to evaluate and experiment very large scale wireless IoT

technologies ranging from low level protocols to advanced services integrated with Internet. FIT IoT-LABs most important goal is to offer an accurate open access multi-user scientific tool to support design, development, tuning, and experimentation related to IoT. Although this testbed has a high availability of nodes with IEEE 802.15.4 technology, it has recently incorporated the possibility of using LoRaWAN nodes. However, one limitation of this testbed for our purpose is that nodes are deployed in both a defined environment (labs) and physical dispositions.

The objective of this work is to develop a testbench that allows developers fast and controlled tests of LoRaWAN networks in different application environments, in order to iteratively obtain the optimal configuration values of this protocol before performing the deployment of the network.

II. LoRa and LoRaWAN

LoRa (Long Range) is a digital wireless data communication technology for long-range, low-power systems developed by Cycleo (France), acquired by Semtech in 2012. This technology aims to be used in devices requiring a long battery-powered life, where energy consumption is of utmost importance. It is necessary to differentiate between LoRa and LoRaWAN (Long Range Wide Area Network). The first term refers normally to the physical layer implementation, while the second one is a specification of LPWAN that defines a Medium Access Control (MAC) protocol and an architecture for this kind of networks.

LoRa is a proprietary spread spectrum modulation scheme derived from Chirp Spread Spectrum (CSS) modulation, in which the signal is modulated by *chirp* pulses (sine-wave pulses of variable frequency). LoRa trades data rate for sensitivity within a fixed channel bandwidth improving resilience and robustness against interference and Doppler effect [6], [7], [8], [9]. It implements a variable data rate, utilizing orthogonal spreading factors, which allows system designers to trade data rate for range. It provides low-power operation, a raw maximum data rate of 27 Kbps (50 Kbps by using Frequency Shift Keying, FSK) and long range (2-5 km in urban areas and up to 15 km in suburban areas). LoRa operates in ISM (Industrial, Scientific and Medical) bands, such as 433, 868 (Europe) and 915 (United States) MHz, depending on the region [10]. Payload size can be defined between 11 and 242 bytes (United States band).

In Argentina, LoRa can be used with the same frequency plan as in Australia, with channels from 915 to 928 MHz divided into the following channel plans (Figure 1).

- $Upstream$: 64 channels numbered 0 to 63 utilizing LoRa 125 kHz BW varying from 0 to DR5, using coding rate 4/5, starting at 915.2 MHz and incrementing linearly by 200 kHz to 927.8 MHz
- $Upstream$: 8 channels numbered 64 to 71 utilizing LoRa 500 kHz BW at DR6 starting at 915.9 MHz and incrementing linearly by 1.6 MHz to 927.1 MHz

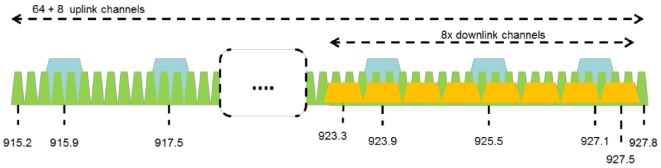

Fig. 1. 915-928 uplink and downlink channels.

TABLE I
DATA RATE CONFIGURATIONS

Data rate	Configuration	Physical bit rate [bits/sec]
0	LoRa:SF12 / 125 kHz	250
1	LoRa:SF11 / 125 kHz	440
2	LoRa:SF10 / 125 kHz	980
3	LoRa:SF9 / 125 kHz	1760
4	LoRa:SF8 / 125 kHz	3125
5	LoRa:SF7 / 125 kHz	5470
6	LoRa:SF8 / 500 kHz	12500
7	RFU*	
8	LoRa:SF12 / 500 kHz	980
9	LoRa:SF11 / 500 kHz	1760
10	LoRa:SF10 / 500 kHz	3900
11	LoRa:SF9 / 500 kHz	7000
12	LoRa:SF8 / 500 kHz	12500
13	LoRa:SF7 / 500 kHz	21900
14	RFU*	
15	Defined in LoRaWAN	

*RFU: Reserved for future use

- $Downstream$: 8 channels numbered 0 to 7 utilizing LoRa 500 kHz BW at DR8 to DR13) starting at 923.3 MHz and incrementing linearly by 600 kHz to 927.5 MHz

Data Rate (DR) is defined as a function of a combination of Spreading Factor (SF) and available Bandwidth (BW), as is shown in Table I. Bit rate can be estimated as is shown in Equation 1.

$$Rb = SF * \frac{1}{\frac{2^{SF}}{BW}} \tag{1}$$

In LoRa modulation, the higher the value of Spreading Factor (SF), the lower the transmission speed, the greater the time of occupation of the transmission in the air, and the reception sensitivity increases, obtaining a more robust communication.

LoRaWAN is an access control protocol, maintained by the *LoRa Alliance*, that allows managing the communication between Gateways and end devices. LoRaWAN networks are organized in a *star of stars topology*, in which there are Gateways that relay messages between nodes and a central Network Server. End-nodes send data to Gateways through a one-hop wireless link and Gateways are connected to the Network Server through standard Internet Protocol (IP) connections. The communication is bidirectional, although uplink communication from nodes to the Network Server is strongly favored, since this can occur at any time, while in most usual cases, downlink communication can only be carried out immediately after an uplink communication, in order to keep

the electrical consumption of nodes as low as possible when the radio is activated for a short time.

III. Network Architecture

The LoRa Alliance is an open and non-profit association initiated by industry leaders to standardize LPWANs. It proposes a network architecture that includes four components: $Nodes$, $Gateways$, $Network\ Servers$ and $Application\ Server$.

This basic architecture works in the following way: Nodes communicate with Gateways using LoRa with LoRaWAN. Gateways forward LoRaWAN frames sent from nodes to a Network Server through another communication interface with a higher throughput, such as WiFi, Ethernet or 3G. This is usually done through software running in the Gateway, called $packet\ forwarder$ which forwards data packets to a Network Server through an IP/UDP (Internet Protocol/User Datagram Protocol) link, and forwards data packets sent by Network Server to end devices. If the Network Server resides in the cloud, the Gateway must have an Internet connection. The Network Server is responsible for routing to the corresponding Application Server, where packets sent by nodes are decoded and packets that must be sent back to end devices are generated. Within this scheme, three classes of node operation are defined, which differ only with respect to the planning of the downlink transmissions:

- *Class A:* supports basic functionality of LoRaWAN, mandatory for all devices. It allows bidirectional communication organized in the following way: for uplink transmissions, devices use non-slotted random access, similar to ALOHA (Additive Links On-line Hawaii Area) medium access method. Download transmission can occur immediately after a successful uplink transmission during two *receive windows*. If the network load is low, class A provides the lowest energy consumption for the nodes, since they only use the radio for a short time.
- *Class B:* implements bidirectional communication with previously planned downlink reception windows. The distribution of information about planning is carried out through $beacons$ sent to nodes by the Gateway.
- *Class C:* devices listen to the channel continuously, providing the lowest latency in downlink, at the expense of an extremely high energy consumption.

IV. Hardware

Most commercially available LoRaWAN nodes use Semtech SX127X radio transceivers, whose main characteristics of different manufacturers are shown in Table II. The main differences about these platforms are *processing and memory capacity* and *programming method*. The appropriate selection of the platform to be used to conform a LoRaWAN network is dependent on the particular application and the developer's knowledge of both programming language and processor type. A detailed comparison of the main characteristics of available LoRaWAN nodes is shown in Table II.

V. Network servers

In this section, we describe the most extended implementations of LoRa Network Servers available at the moment.

A. The Things Network, TTN

The Things Network (TTN) is a project to build a decentralized IoT data network, open and funded by the community, where users are its owners and operators [12]. It started in Amsterdam (Netherlands) and, currently, TTN connected Gateways can be found all over the world, including Argentina.

TTN uses LoRaWAN, and covers the entire stack: libraries for devices, software for Gateways, cloud routing services, software development kits, integrations, etc. Its architecture (Figure 2) is designed to be scalable and distributed, allowing high availability, high performance and end-to-end security, complying to the LoRaWAN specification. The core components are the following:

- Node: LoRaWAN end device.
- Gateway: device that, in uplink communication, receives LoRa messages and forwards them to the Router. In downlink, it forwards packets received from the Router to the nodes.
- Router: microservice that, in uplink communication, receives messages from the Gateway and finds a Broker to forward the message to. It is responsible for all functionality linked to the Gateways and the specific details of the operating region. In downlink, it is responsible for planning and forwarding packages received from a Broker to the corresponding Gateway just-in-time.
- Broker: microservice that, in uplink communication, identifies the device within a range of addresses, deduplicates the traffic and forwards the packet to the Handler where the application is registered. In downlink, it forwards the messages received from the Handler to the corresponding router.
- Handler: microservice that, in uplink communication, decrypts the packet payload and forwards the messages to the applications. In downlink, it encrypts the payload received from an application and forwards it to the corresponding Broker.
- Application: software running on a server, performing some tasks for the end user.

In addition to these core components, there are two complementary modules: *Discovery Server*, which keeps track of different components that are registered in the network, and *Network Server*, which contains the registry of devices present in the network and monitors their state. The appropriate selection of the platform to be used is dependent on the particular application.

B. Orbiwise

OrbiWAN of Orbiwise [13] is a leading solution provider for LPWAN IoT optimized for connected objects based on LoRa technology. OrbiWAN is a fully compliant to LoRa 1.0.2 with support for all features of the specification and all devices class with A, B and C. This is a commercial solution with

TABLE II
LORAWAN NODES

Parameter / Node Model	Semtech MultiConnect mDot	Microchip RN2903 Node	Pycom LoPy v4
Semtech Radio Transceiver	SX1272	SX1276	SX1276
LoRa Compatibility Version	v1.0.2	v1.0.2	v1.0.2
Supported Operation Classes	A, C	A, C	A,C
Gateway Functionality	No	No	Yes (WiFi)
Processor	32-bits ARM Mbed STM32F411RET	8-bits 18LF46K22 (radio), PIC18LF45K50 (host)	32-bits ESP32 Dual Core
Flash Memory	512 KB (400 KB usable)	32 KB	8 MB
RAM	128 KB	2 KB	4 MB
Programming Language	C	C	MicroPython
AT Commands	Yes	Yes	No
Use Complexity	Medium	Medium	Low
Release year	2016	2016	2017
Price w/antenna (USD)	70	69.99	61.50

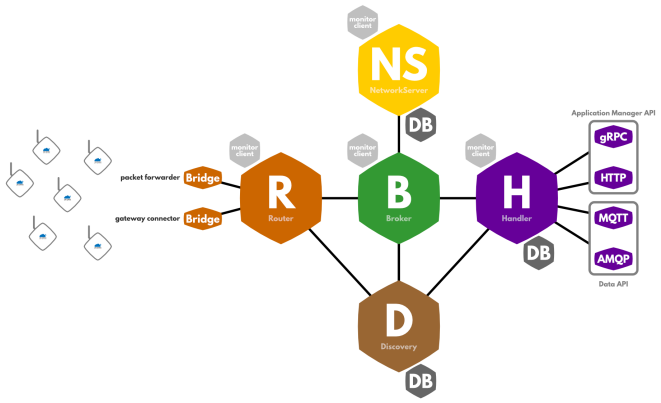

Fig. 2. TTN Architecture.

little information available. Therefore, it has some limitations in the current implementation, which limit, for example, the maximum payload size to very low values. This limits some task such as firmware update of the nodes in the network, since it takes a lot of time to perform this task. Likewise, this network server implementation is responsible for carrying out all network management, allowing the user to only send and visualize data received by the network server, and abstracting it from the management of intermediate stages.

C. Lora Server

The LoRa Server project provides open source components for building LoRaWAN networks, covering all the necessary services between gateways that receive messages from nodes until just before applications that receive data [14]. It provides mechanisms to manage gateways in the LoRa network, the supported applications and devices associated with the applications. The architecture proposed by LoRa Server consists of multiple components (Figure 3):

- *Node:* device that sends data to the LoRa network.
- *Gateway:* device that forwards data from nodes to the LoRa Network Server and vice versa.
- *LoRa Gateway Bridge:* component responsible for communication with the Gateway. It transforms from UDP with which packets are sent from the Gateway to JSON through MQTT (Message Queuing Telemetry Transport).
- *LoRa Server:* component responsible for controlling the network. It knows the active sessions of nodes and, when a new node tries to join the network, it will check with the application server to know if it is allowed in the network and, if so, verify what configuration to use for that node. It de-duplicates the received data, authenticates it and forwards it to the Application Server and, if necessary, will return response packets to the sender node.
- *LoRa App Server:* component that implements an Application Server compatible with the LoRa Server. It offers administration of nodes by application and by organization, and administration of Gateways by organization. It also offers user administration and the possibility of assigning them to organizations and/or applications. Communication with the applications is done through JSON (JavaScript Object Notation) over MQTT and using Application Programming Interfaces (APIs).
- *Application:* software that receives data from nodes and sends them replies.

VI. IMPLEMENTATION

In this section, we analyze and compare the features of the main available LoRaWAN Network Servers. First, the implemented system is described, and then the LoRa Server implementation is detailed.

A. Implemented system

The hardware used on the final implementation of the system was the following:

- *Nodes:*
  1 Pycom LoPy v4
  1 Microchip RN2903
  1 Multitech mDot
- *Gateway:* 1 Multitech Conduit
- *Network Server:* LoRa Server

B. Network Server Selection

The second most important component to be defined of a LoRa network is the network server. This component is

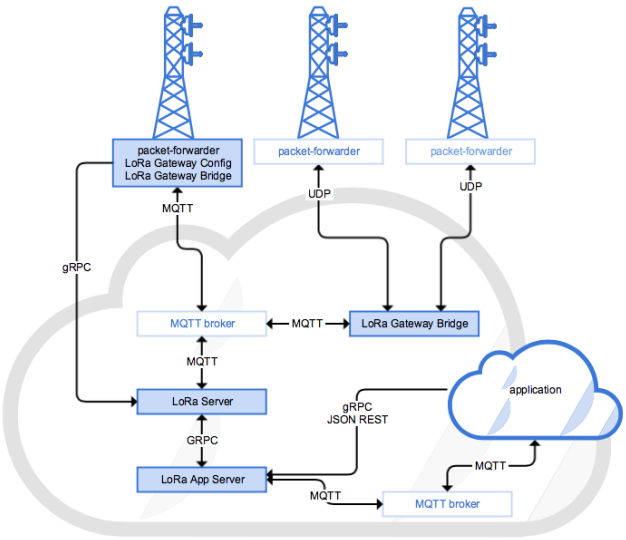

Fig. 3. LoRa Server architecture.

responsible for managing all components in the network. Table III shows the main features of different network servers currently available. The option chosen initially was TTN but, due to security restrictions in the laboratory network (TTN used UDP 1700 port), we opted to change to LoRa Server to perform the tests.

C. Network Server installation

To install LoRa Server we used a virtual machine with Ubuntu 16.04 LTS, implementing a scheme such as the one outlined in the installation guide that appears on the project's website, in which all the components and their dependencies are installed in a single instance of the server, including the LoRa Gateway Bridge, which can optionally be installed on the same Gateway. As a prerequisite, the following software must be installed:

- MQTT Broker: as mentioned above, MQTT is a messaging protocol of publication/subscription. Allows users to publish information within topics to which others can subscribe. In this case, the broker used is Mosquitto.
- *Redis:* database used to store relatively transient data.
- *PostgreSQL:* database for long-term storage.

Once the software is configured and the databases and their users are created, we proceed to the installation of the different modules that make up LoRa Server. The repository that contains the necessary files is available at https://repos.loraserver.io/ubuntu/. This repository was added to the list of repositories of the APT package manager that uses Ubuntu and, in this way, can use this manager to download and install the modules. Once the installation is complete, you can proceed to configure each module.

D. Lora Gateway Bridge

The LoRa Gateway Bridge configuration file can be found in /etc/lora-gateway-bridge/lora-gateway-bridge.toml. The most important fields within it for the initial configuration are the following:

- packet_forwarder.udp_bind: interface and port on which LoRa Gateway Bridge receives packages from the Gateways. It must be 0.0.0.0:1700 to allow access from all network interfaces
- backend.mqtt.username, backend.mqtt.password: due to the fact that the MQTT server is publicly accessible, it is convenient to use a username and password. These credentials are entered when configuring Mosquitto.

E. Lora Server

The LoRa Server configuration file can be found in /etc/loraserver/loraserver.toml. The most important fields within it for the initial configuration are:

- network_server.band.name: ISM band used. In this case, the american band was used, so its value must be US_902_908.
- postgresql.dsn: URL of the Postgre server.
- network_server.gateway.backend.mqtt.username, network_server.gateway.backend.mqtt.password: credentials for configuring Mosquitto (same as in the LoRa Gateway Bridge configuration).
- network_server.gateway.api.jwt_secret: a secret value used to generate gateway tokens. It can contain any value.

F. Gateway configuration

After the installation of the network server, the next step was to configure the gateway. That implies configuring the packet forwarder that runs in the Gateway to forward the packets to and from the Network Server. By default, the Gateway comes with a demonstration with the packet forwarder that forwards packets to a server of the manufacturer (Semtech) in the cloud. Additionally, the Gateway has a basic Network Server to perform initial tests. This server should be disabled and the packet forwarder must be reconfigured to resend the packets to the new Network Server (LoRa Server) that has been installed in the previous section.

VII. Evaluation

The three node models were configured to send data packets to an application in the LoRa Server at regular time intervals. (Figure 4). A Multitech Conduit Gateway was used in which a LoRa packet forwarder software process is running in order to forwards RF packets receive by the concentrator to a server through a IP/UDP. Nodes were used in Class A mode and performed an Over-The-Air Activation (OTAA) to join the network. Once nodes are accepted, they would send test packages with information. To corroborate the correct functioning of both uplink and downlink, it was decided to use confirmed transmissions. This implies that, when a packet from a node is received, if the packet was received correctly, a

TABLE III
LoRaWAN Network Servers

Parameter / Implement.	TTN	OrbiWAN	LoRa Server
Support	1.1	1.0.2	1.0.2, 1.1 (Q3 2018)
LoRa v1.1 Support	yes	B Class	no
License	Open Source, MIT License	Commercial	Open Source, MIT License
Language	Go	-	Go
Dev Community	high	low	medium
Release Year	2015	2015	2017

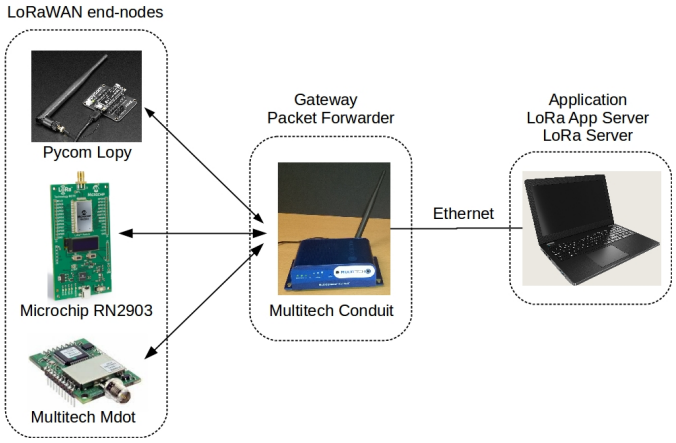

Fig. 4. Implemented testbench architecture.

confirmation message (acknowledgment or ack) must be sent back to it. Fragments of a network server log are shown below:

```
mar 08 13:48:54 loraserver loraserver[974]:
time="2018-03-08T13:48:54-03:00"
level=info msg="gateway updated"
mac=240ac4fffe00be7a

mar 08 13:49:13 loraserver loraserver[974]:
time="2018-03-08T13:49:13-03:00"
level=info msg="rx info sent
to network-controller"
dev_eui=0004a30b001ec7bd

mar 08 13:49:13 loraserver loraserver[974]:
time="2018-03-08T13:49:13-03:00"
level=info msg="device updated"
dev_eui=0004a30b001ec7bd
```

During the evaluation of the system, different types of problems arose. For example, not all nodes were configured by default to use the same LoRa frequency channels that the Gateway can listen to and that the Network Server was configured to use. Once this problem had been solved by modifying configuration parameters, it was possible to start with real communication tests. Nodes were evaluated using different packet forwarding times (between 1 and 70 seconds). In this sense, it was verified that to transmit packets every few seconds (around 1-10 seconds) there was a packet loss probability of 20% mainly due to delays in packet processing tasks in packet forwarder and in Network Server. In the first case, the packet queue of the packet forwarder running at the Gateway is very small, so when a new packet arrives after a short time it can overwrite an earlier packet that has not yet been processed or forwarded.

VIII. Conclusions

In this work, we introduce the main characteristics of LoRa and LoRaWAN wireless communication technology, a comparison of the main features of hardware and network servers, initial implementations and the difficulties detected. After the implementation of a LoRaWAN testbench based on LoRa Server, we observed a very good stability, compatibility between nodes of different manufacturers, but that good knowledge of the protocol is required to configure some critical parameters of the physical and link layer to achieve stable operation. Future work considers the centralized acquisition and off-line processing of logs obtained from each element of the Lora Server network for analysis of communication performance and reconfiguration of LoRaWAN's parameters.

References

[1] D. Bankov, E. Khorov, and A. Lyakhov, "On the Limits of LoRaWAN Channel Access," IEEE International Conference on Engineering and Telecommunication, 2016

[2] O. Georgiou, and U. Raza, "Low Power Wide Area Network Analysis: Can LoRa Scale?," IEEE Wireless Communications Letters, vol. 6, no. 2, pp. 162-165, April 2017. doi: 10.1109/LWC.2016.2647247

[3] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Meli-Segu, and T. Watteyne, "Understanding the Limits of LoRaWAN," IEEE Communications Magazine, 2017.

[4] Q. Lampin, and D. Barthel, "Sensorlab2: A Monitoring Framework for IoT Networks, " International Conference on Performance Evaluation and Modeling in Wired and Wireless Networks (PEMWN), 2017.

[5] C. Adjih, E. Baccelli, E. Fleury, G. Harter, N. Mitton, T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele, and T. Watteyne, "FIT IoT-LAB: A Large Scale Open Experimental IoT Testbed," In proc. 2nd World Forum on the Internet of Things (WF-IoT), 2015.

[6] Semtech Corp, "AN1200.22 LoRa Modulation Basics," May 2015. Available at: http://www.semtech.com/images/datasheet/an1200. 22.pdf

[7] A. Augustin, J. Yi, T. Clausen, and W. M. Townsley, "A Study of LoRa: Long Range & Low Power Networks for the Internet of Things, " MDPI Journal Sensors 2016, 16, 1466; doi:10.3390/s16091466, 2016.

[8] "LoRaWAN 1.0.2 Specification," 2016.

[9] "LoRaWAN 1.1 Specification," 2017.

[10] "LoRaWAN 1.1 Regional Parameters," 2017.

[11] B. Kim, and K. Hwang, "Cooperative Downlink Listening for Low-Power Long-Range Wide-Area Network," MDPI Journal Sustainability 2017, 9, 627; doi:10.3390/su9040627, 2017.

[12] The Think Network, available at: https://www.thethingsnetwork.org/

[13] Orbiwise, available at: https://www.orbiwise.com/home

[14] Lora Server, available at: https://www.loraserver.io/

[15] C. A. Boano, S. Duquennoy, A. Frster, O. Gnawali, R. Jacob, H. Kim, O. Landsiedel, R. Marfievici, L. Mottola, G. P. Picco, X. Vilajosana, T. Watteyne, and M. Zimmerling, "IoTBench: Towards a Benchmark for Low-power Wireless Networking," Proceedings of the First Workshop on Benchmarking Cyber-Physical Networks and Systems, 2018.