Building a Private LoRaWAN Platform

John J. Lee, Youssef Souryal, Darren Tam, Dongsoo Kim, Kyubyung Kang, Dan D. Koo

Abstract: LoRaWAN technology has been here for several years as one of LPWAN technologies. It consists of various components such as end nodes, a gateway, a network server, and an application server at the minimum. The servers have been exclusive products of commercial companies, and not many experimental or academic ones are available. Recently one such software has been developed. However, few fully functional academic ones have been reported. In this study, we implement a fully functional private independent LoRaWAN platform for the academic research of LPWAN Internet of Things (IoT) and demonstrate that our platform can support not only end-to-end LoRaWAN communication but also graphical user interface on an embedded and limited computing power system.

Keywords : LoRaWAN, LPWAN, Gateway, Network Server, Application Server.

I. INTRODUCTION

The rapid development of electronics technology expects to deploy 20 billion IoT devices by 2020. To connect such a tremendous number of devices, several low power wide area (up to several mile ranges in line of sight) networking (LPWAN) technologies have been developed, which include LoRaWAN [1], Sigfox [2], Wi-SUN [3], etc. As compared to other LPWAN technologies, LoRaWAN has the following advantages. (i) It is one of the most widely adopted long range wireless communication technologies for Internet-of-Things (IoT) using an ISM radio band. (ii) It has star-of-star topology, and thus, it is easier to debug and isolate faults. (iii) It has an extended battery lifespan. Therefore, the technology has been utilized for building smart and safe urban infrastructure, smart farms, smart disaster warnings and preparedness, etc.

Although LoRaWAN is quite popular to many IoT developers and users, and thus, several LoRaWAN platforms have been deployed by commercial companies, there is lack of academic or research LoRaWAN end-to-end platforms except The Things Network (TTN [4]), and no report of private independent (stand-alone) non-commercial platforms to our knowledge.

This article describes a private LoRaWAN platform that can be used for academic and research purposes. This platform has been built on a Raspberry Pi 3 64-bit quadcore processing system utilizing ChirpStack.io project [5] but can be extended to any number of gateways and servers.

II. PRIOR WORK

LoRaWAN technology has been developed and deployed in 2012, but most often network service providers offer the infrastructures such as Comcast in the US, KPN in Europe, and SK Telecom in Asia. There exists only one non-commercial but public LoRaWAN service provider called The Things Network (TTN [4]). Unlike Sigfox (another LPWAN technology), anyone can build and deploy any of LoRaWAN gateways and servers, which is a big advantage of using LoRaWAN. Nonetheless, as this technology is quite new, end-to-end private independent LoRaWAN platforms are very rare. Per our knowledge, there is only one software project (ChirpStack.io [5]) which implements open-source LoRaWAN server-side software and can be used to build a LoRaWAN platform. Moreover, it is not well known in the US, and thus, there is no role model of US version of private LoRaWAN platforms.

III. METHODOLOGY

A. Problem Statement

Fig. 1 shows the basic LoRaWAN platform architecture. To build its whole service system, all of the components in the figure must be realized. Except for end-nodes, such components are typically provided by a few commercial service providers. Thus, users need to pay subscription fees and rely on them, but the worst is the privacy issue as the providers have users' data. For the applications that are necessary to keep private, or those who want/need to build private LoRaWAN platforms, a solution is necessary. Hence, embedded LoRaWAN platforms can be very beneficial to small private or academic projects such as IoT courses, small farms, home automation, and small-size buildings.

B. The Overall Structure of our Private LoRaWAN Platform

Our LoRaWAN platform has a layered structure consisting of four layers: a device layer, a network layer, an information layer, and an analysis layer as shown in Fig. 2.

Device Layer: Each device consists of a microprocessor, wireless communication modules, sensors and/or actuators, and software and/or operating system. If support to the operating system is required for edge computing or somewhat complex tasks, the microprocessor should be 32-bit or higher processors such as ESP32 or ARM derivatives. Several operating systems (OS) or Real-Time OS (RTOS) have been developed for IoT such as

Revised Manuscript Received on December 16, 2019.
* Correspondence Author

John Lee*, **Youssef Souryal**, **Dongsoo Kim, Kyubyung Kang, Dan D. Koo**, ECE department, IUPUI, Indianapolis, USA. Email: {johnlee, ysouryal, dskim, kyukang, dankoo}@iu.edu

Darren Tam, Tandon School of Engineering, NYU, Brooklyn, USA, dt1453@nyu.edu

This research has been partly supported by the NSF Award #1757636, the STEM-SSI (Summer Scholars Institute), and Multidisciplinary Undergraduate Research Institute (MURI) as well as Research Support Funds Grant (RSFG) of IUPUI.

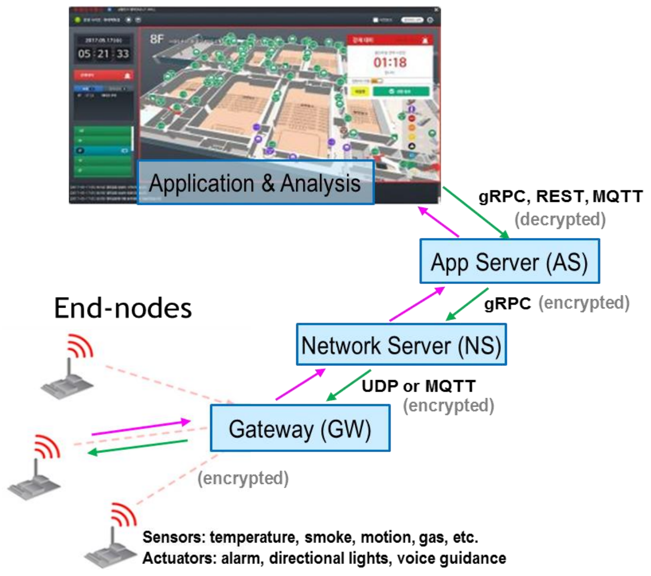


Fig. 1. Private LoRaWAN platform architecture.

FreeRTOS [6] or Contiki OS [7]. FreeRTOS is widely accepted in IoT development due to its small footprint, scalability, extensive support for a wide spectrum of hardware architecture.

Network Layer: This layer consists of the following connections as shown in Fig. 1: (i) Communication between devices and gateways, and (ii) Communication between gateways and network servers and application servers. The design objectives of the network layer include (i) reliable communication with fault tolerance, (ii) secure communication, (iii) covering entire building including basements, (iv) easiness to program, test, and upgrade the software, and (v) standalone operation (independent of external service providers). One advantage of LoRaWAN is that one gateway (GW) can cover an entire mid-size building. However, for redundancy and reliability, typically more than one GWs are installed.

Information Layer: This layer gathers all kinds of information from sensors such as temperature, humidity, human existence, position, acceleration, timestamps, and motions depending on applications. Then it should categorize the types of data based on properties of sensors so that it can serve the analysis layer and applications appropriately and efficiently. This layer should also have application specific information such as building structures and floor plans for smart buildings, land sections in smart farms, or street shapes/maps in smart cities.

Analysis Layer: This layer utilizes all the information provided by the network and information layers. Synthesizing multiple source information over time extracts

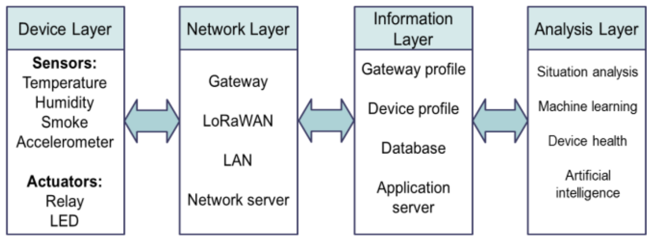


Fig. 2. Private LoRaWAN platform layers.

knowledge and understands the context and situation. For smart cities, as an example, information from signal traffic loops would be fused with information of vehicle built-in devices via road-side units to understand the impact of a car accident and mechanic problem to traffic flow. This layer should be able to make situation-aware or context-aware decisions. To extract valuable information out of the enormous data stemming from various IoT sources, it can exploit machine learning and/or artificial intelligence to learn and deduce abnormal situations.

IV. IMPLEMENTATION

In our implementation, we have set up three layers of our private LoRaWAN platform, namely, the device layer, network layer, and information layer. The implementation of the analysis layer has been postponed as it is fairly sophisticated.

A. The Components of a Private LoRaWAN Platform

As a solution to the aforementioned necessities, we build a prototype private LoRaWAN service platform based on ChirpStack project [5] along with other software such as MQTT and gRPC. As this is a sample prototype, for hardware, we have used a Raspberry Pi 3 (RPi3) 64-bit quadcore processor system and a Semtech LoRa concentrator with 9-channel LoRa transceivers, more specifically LoRaGo PORT [8].

The component connection diagram of our platform is shown in Fig. 3. The platform consists of the following software modules: a packet forwarding software module, a gateway bridge module, a network server module, an application server module, an MQTT broker module, and a PostgreSQL database module. All of these software modules are running on the Raspberry Pi 3 (RPi3) in our implementation, but it does not necessarily to be like this as most of the software modules can be spread over separate computers or even in the cloud. Detailed connections are explained in the following subsection.

B. Architecture and Connections of Hardware and Software Components

The so-called gateway (GW) is composed of a LoRa transceiver, a packet forwarding software module, and a bridge software module. The packet forwarding module we use is the Semtech UDP Packet Forwarder [9].

The network server consists of an MQTT broker module and a network server module. The gateway bridge module running on RPi3 passes packets between the packet forwarder and the MQTT broker, which relays again the packets from/to the network server module. MQTT messages are used in this stage due to the following advantages. (i) As this platform is expected to be used in academia, it should be easy to debug the communications. The best way to do this is to use MQTT messages as they are easier to publish and subscribe for logging purpose or even manually triggering events such as controlling actuators or just LEDs. (ii) Packets from different gateways can have different head topics, and thus, messages can be easily routed to different gateways and end devices.

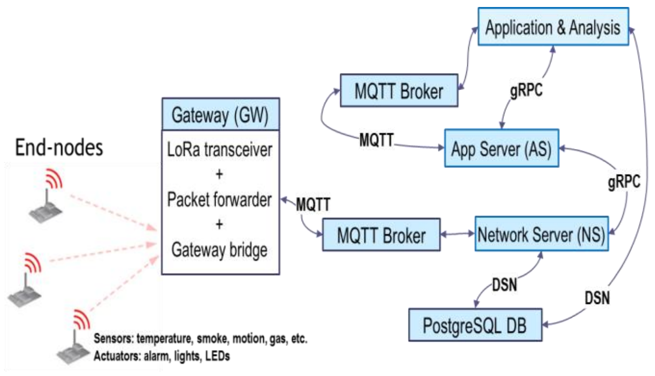


Fig. 3. The connection diagram of our platform.

The Application Server (AS) consists of an application server module and gRPC through which the AS connect to the network server and applications if desired. gRPC [10] is a high-performance, open-source, universal, language and platform independent RPC framework on top of protocol-buffers [11] which are Google's language-neutral, platform-neutral extensible mechanism for serializing structured data – like XML, but smaller, faster, and simpler.

In order to store and later analyze persistent data, PostgreSQL database [12] is used, which is accessed from the network server module as well as application server module using Data Source Name (DSN).

C. LoRaWAN Devices (Sensors and Actuators)

For end devices, we have used three different LoRa boards, namely, SODAQ board [13], Marvin board [14], and STM32 B-L072Z-LRWAN1 board [15]. SODAQ LoRaWAN Explorer [13] includes a LoRa transceiver, a 32-bit ARM microcontroller, a temperature sensor, a multicolor LED for output monitoring, and Arduino M0 Compatible analog and digital inputs.

Marvin board [14] consists of a USB port, an Arduino compatible microcontroller (AtMega 32u4), five Grove connectors, and a Microchip LoRaWAN modem. The combination of these components allows users to very easily connect various analog and digital sensors and actuators, and program using Arduino IDE [16]. We have tested temperature, humidity, and light intensity sensors on this board.

B-L072Z-LRWAN1 Discovery kit [15] with a sensor shield from STM has various sensors such as an accelerometer. However, it requires to use System

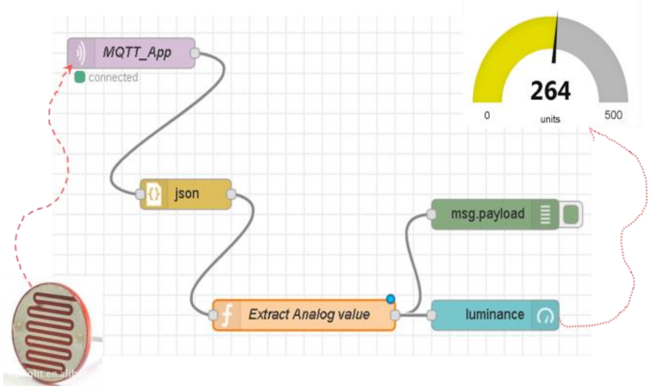
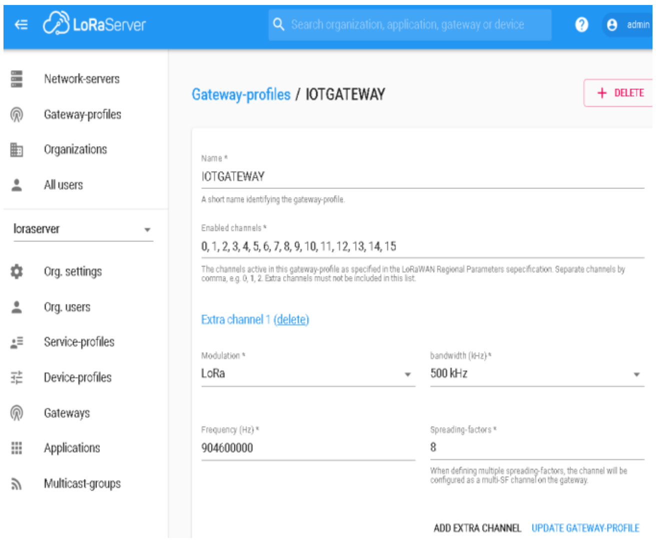


Fig. 5. One of experimental applications.



Gateway-profiles / IOTGATEWAY

Name *
IOTGATEWAY
A short name identifying the gateway-profile.

Enabled channels *
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
The channels active in this gateway-profile as specified in the LoRaWAN Regional Parameters sepecification. Separate channels by comma, e.g. 0, 1, 2. Extra channels must not be included in this list.

Extra channel 1 (delete)

Modulation *
LoRa

bandwidth (kHz) *
500 kHz

Frequency (Hz) *
904600000

Spreading-factors *
8
When defining multiple spreading-factors, the channel will be configured as a multi-SF channel on the gateway.

Fig. 4. Gateway profile.

Workbench IDE [17] from STM or other sophisticated IDEs for development such as Mbed [18], which makes beginners hard to approach at first, but is preferred for experts.

D. Configuration of a Private LoRaWAN Platform

The various software modules need to be configured to properly set numerous parameters for meeting not only the LoRaWAN specification but also the US FCC regulation for the usage of the platform in the US. To do that, we set up three device profiles (for three device types), a gateway profile and its service profile, and configure the modules to be abided by the US FCC regulations.

For example, Fig. 4 shows "Gateway Profile," which selects which channels the gateway will use. For Comcast service, channels 0 to 7 are used, whereas for TTN service, channels 8 to 16 are used.

E. Applications

For applications, we explore a few end devices (nodes) with various sensors on three different afore-described boards. The sensor data includes temperature, humidity, barometer, accelerometer, and light intensity as an analog signal, and they are sent to the gateway, network server, and application server.

For application GUI, we use Node-RED [19] tool with its Dashboard features as they are very easy to integrate. For application data formatting, we used Cayenne LPP [20], which is de facto IoT data formatting standard that enables coherent data sharing among servers and applications for seamless interoperability.

V. EXPERIMENTATION

A. Experimental Setup

We have integrated all the aforementioned hardware and software components on a Raspberry Pi 3 with a Semtech LoRa concentrator shield [8].

We have experimented several applications. Fig. 5 shows an application where the end node simply measures light intensity and transmits to GW, which relays the data to the network and application servers, and then to the application, i.e., Node-RED Dashboard.

B. Experimental Results

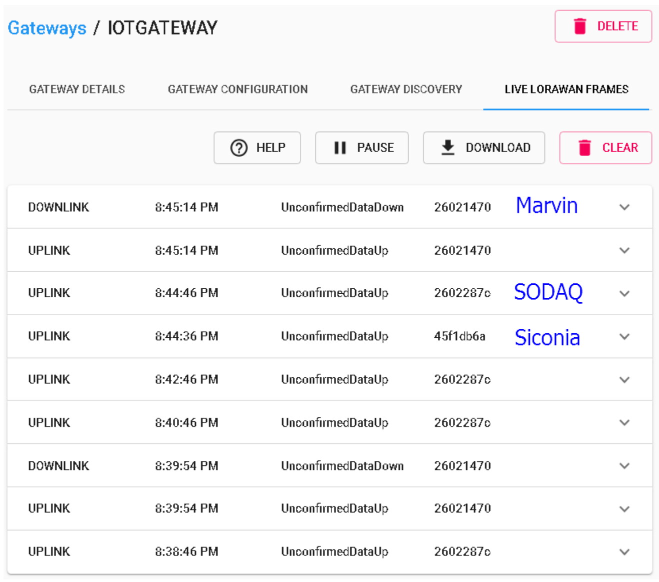
Fig. 6 shows communication status from three devices to the gateway. Packet details of uplinks and downlinks' messages can be revealed by clicking v (down arrow) on the right side.

Fig. 7 shows the MQTT message at the gateway received from a sensor node. It is therefore easy to check and debug communication status. Notice that the application packet data (denoted as phyPayload) is encrypted, which is according to the LoRaWAN specification. However, for testing purpose, one can use un-encrypted packets.

Fig. 8 shows the MQTT message at the application server side that is originated from the Marvin board. It also shows very detailed information about the application-side states. As the application server can now decrypt the payload and also the payload format used is Cayenne LPP data representation [18], the application server interprets and neatly presents three different types of sensor data in JSON format, which can be easily parsed by a JSON parser Node-RED node. Then, data can be exhibited by the Node-RED and its Dashboard as shown in Fig. 5.

VI. CONCLUSION

We have successfully built a private independent non-commercial LaRaWAN platform that does not require any external service providers. This platform can be used for various applications such as smart city, automated factory, smart farm applications, natural disaster warning and preparedness prediction, infrastructure resilience and condition monitoring, and damage mitigation. It can also be easily extended to bigger platforms as more end-nodes are integrated and data are accumulated rapidly. This is possible by building separate application servers as gRPC ([10],


Gateways / IOTGATEWAY

DOWNLINK	8:45:14 PM	UnconfirmedDataDown	26021470	Marvin
UPLINK	8:45:14 PM	UnconfirmedDataUp	26021470	
UPLINK	8:44:46 PM	UnconfirmedDataUp	2602287c	SODAQ
UPLINK	8:44:36 PM	UnconfirmedDataUp	45f1db6a	Siconia
UPLINK	8:42:46 PM	UnconfirmedDataUp	2602287c	
UPLINK	8:40:46 PM	UnconfirmedDataUp	2602287c	
DOWNLINK	8:39:54 PM	UnconfirmedDataDown	26021470	
UPLINK	8:39:54 PM	UnconfirmedDataUp	26021470	
UPLINK	8:38:46 PM	UnconfirmedDataUp	2602287c	

Fig. 6. Monitor outputs of communication status at the gateway.

```
gateway/B827EBFFFE------/tx
{"token":11094,"txInfo":{"mac":"B827EBFFF
E------","immediately":true,"frequency":9
15000000,"power":20,"dataRate":{"modulati
on":"LORA","spreadFactor":12,"bandwidth":
500},"codeRate":"4/5","iPol":false},"boar
d":0,"antenna":0},"phyPayload":"IMaBI8exY
8t1KqOGgvaM/FM="}
```

Fig. 7. MQTT message at the gateway.

shown in Figures 1 and 3) can communicate across LAN or Internet.

```
application/5/device/0004a30b0021f8fe/rx
{"applicationID":"5","applicationName":"MARVI
N_APP", "deviceName":"MARVIN",
"devEUI":"0004a30b0021f8fe",
"rxInfo":[{"gatewayID":"b827ebxxxe795a26",
"name":"IOTGATEWAY","time":"2019-03-25T00:50:
46.229716Z", "rssi":-22,
"loRaSNR":8.8,"location":{"latitude":39.96756
,"longitude":-85.92969,"altitude":261}}],
"txInfo":{"frequency":905300000,"dr":0},"adr"
:true,"fCnt":0,"fPort":2,"data":"AQIbRANnANIF
aEA=", "object":{"analogInput":{"1":71.6},
"temperatureSensor":{"3":22},"humiditySensor"
:{"5":33}}}
```

Fig. 8. MQTT message at the application server side.

For future work, we plan to build a bigger platform on a server running an Ubuntu operating system.

REFERENCES

1. LoRa Alliance, Available: https://lora-alliance.org/
2. Sigfox, Available: https://www.sigfox.com/en
3. Wi-SUN, Available: https://www.wi-sun.org/
4. The Things Network (TTN), Available: https://www.thethingsnetwork.org/
5. ChirpStack, Available: https://www.chirpstack.io/, 2019.
6. FreeRTOS, Barry, Richard. "FreeRTOS," Internet, Oct (2008), https://www.freertos.org/
7. A. Dunkels, B. Gronvall, and T. Voigt, "Contiki-a lightweight and flexible operating system for tiny networked sensors," 29th annual IEEE international conference on local computer networks, pp. 455-462, 2004.
8. LoRaGo PORT, Available: https://sandboxelectronics.com/?product=lorago-port-multi-channel-lorawan-gateway
9. Semtech UDP Packet Forwarder, Available: https://github.com/Lora-net/packet_forwarder
10. gRPC, https://grpc.io/
11. Google, "protocol-buffers," Available: https://developers.google.com/protocol-buffers/
12. PostgreSQL, Available: https://www.postgresql.org/
13. SODAQ, Available: https://support.sodaq.com/sodaq-one/explorer/
14. Marvin, Available: https://github.com/iotacademy/marvin
15. STM32 B-L072Z-LRWAN1, Available: https://www.st.com/en/evaluation-tools/b-l072z-lrwan1.html
16. Arduino IDE, https://www.arduino.cc/en/Main/Software
17. System Workbench IDE, Available: https://www.st.com/en/development-tools/sw4stm32.html
18. ARM Mbed, Available: https://www.mbed.com/en/
19. Node-RED, Available: https://nodered.org/
20. Cayenne LPP, Available: https://community.mydevices.com/t/cayenne-lpp-2-0/7510

AUTHORS PROFILE

John J. Lee, Associate Professor, Department of ECE, IUPUI, USA. He received his Ph.D. degree in Electrical and Computer Engineering from the Georgia Institute of Technology. http://www.ece.iupui.edu/~johnlee/

Youssef Souryal, a graduate student at Purdue School of Engineering and Technology, Indianapolis. He graduates in December 2020 with master's degree. Youssef is passionate about software engineering.

Darren Tam, Electrical engineer graduated from NYU Tandon School of Engineering. His passion lies in the energy and communication fields because he believes that in the future, we will be able to live in a more energy efficient and technologically connected society.

Dongsoo S. Kim, Associate Professor, Department of ECE, IUPUI, USA. He received his Ph.D. degree in Computer Science and Engineering from the University of Minnesota. http://et.engr.iupui.edu/~dskim

Kyu Kang, Assistant Professor, Engineering Technology, Construction Management, IUPUI, He received his Ph.D. degree in Civil Engineering from the Purdue University, https://et.iupui.edu/people/kyukang

Dan Koo, Associate Professor, Construction Management program at IUPUI. He is a registered PE in Civil Engineering and received a Ph. D. in Civil and Environmental Engineering from Arizona State University. https://et.iupui.edu/people/dankoo