Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Design and Implementation of Open LoRa for IoT

QIHAO ZHOU[1], KAN ZHENG[1] (Senior Member, IEEE), LU HOU[1], JINYU XING[1], AND RONGTAO XU[2],

[1]Intelligent Computing and Communication (IC^2) Laboratory, Key Laboratory of Universal Wireless Communications, Ministry of Education, Beijing University of Posts & Telecommunications, Beijing, China, 100876

[2]State Key Laboratory of Rail Traffic Control and Safety, Beijing Jiaotong University, Beijing, China, 100044

Corresponding author: K. ZHENG (zkan@bupt.edu.cn)

ABSTRACT Long Range (LoRa) network is emerging as one of the most promising Low Power Wide Area (LPWA) networks, since it enables the energy-constraint devices distributed over wide areas to establish affordable connectivity. However, how to implement a cost-effective and flexible LoRa network is still an open challenge. This paper aims at exposing a feasible solution of design and implementation, allowing users to conveniently build a private LoRa network for various IoT applications. Firstly, several typical application scenarios of LoRa network are discussed. Then, the LoRa system architecture is presented with the functionality of each component. We address the hardware design and implementation of LoRa Gateway, which is the bridge between LoRa nodes and LoRa network server. Especially, the paper contributes by proposing an improved software architecture of LoRa network server whose source codes are open on GitHub. Under the architecture, LoRa network server is divided into four decoupled modules and uses the messaging system based on streaming data for the interaction between modules to guarantee scalability and flexibility. Finally, extensive experiments are conducted to evaluate the performance of LoRa networks in typical environments.

INDEX TERMS LoRa, LPWA, IoT, Microservice, Open Source

I. INTRODUCTION

With the development of low power wide area (LPWA) technologies, the flourish of various Internet of Things (IoT) applications is changing the way of living in the world. In particular, it is estimated that LPWA devices will grow to 339 million in 2025 [1]. Due to its unique technical features, Long Range (LoRa) network has attracted significant attention from industry and academic sectors in recent years. LoRa operates in the unlicensed Industrial Scientific and Medical (ISM) band and offers the long-distance connectivity to low-power devices. On the other hand, LoRaWAN is proposed by LoRa Alliance, which defines the network protocol in the medium access control (MAC) and network layers [2]. A completed LoRa network primarily consists of several components, *i.e.*, LoRa Node, LoRa Gateway, LoRa network server and LoRa application server. It is worth mentioning that LoRa Nodes usually operate on battery power and have limited computational capabilities. LoRa network server is responsible to have sufficient ability to provide the packet processing services and support massive number of LoRa Nodes. Therefore, the architecture and implementation of the LoRa network server have a significant impact on the capacity of LoRa system.

There has been vast amounts of work being explored for LoRa network. Firstly, some of studies focused on comparing LoRa with other LPWA technologies such as Sigfox and NB-IoT [3]–[5]. The authors in [6]–[8] analyzed and discussed the wide variation of performance in specific scenarios as well as the capabilities and limitations. These studies used analytical and simulation-based approaches to carry out the experiments and gave the processing performance of background server. There are various challenges when deploying LoRa network in practical scenario. Moreover, several IoT cloud platforms have been proposed to process and manage IoT data [9]–[11]. However, LoRa Nodes are distributed over wide areas and utilize LoRa Gateway to establish the connection with Internet. These services provided by IoT cloud platforms are failed to be used directly by LoRa Nodes. A LoRa network server project is necessary to process the packets forwarded from/to LoRa Gateway. In [12], the authors proposed an implementation of LoRa network server on the OpenStack named as LoRaCloud without the performance evaluation. In addition, there are some projects available for building and managing LoRa network. For example, The Things Network (TTN) has begun to release an open source platform for LoRa network [13]. However, independent deployment of TTN-based LoRa network server is cumbersome, which is not conducive to agile development. Another popular open source project is named LoRaServer [14]. LoRaServer presented a ready-to-use solution including design and implementation. But the centralized architecture leads to excessive functionality concentrated on one module. It may have a negative impact on the throughput and scalability of entire system. Therefore, an open LoRa network with high performance and flexibility is highly desirable.

In this paper, the typical application scenarios and requirements of LoRa network are presented firstly. Then, the LoRa system architecture is discussed as well as the functions of each component. In order to reduce the difficulty of deploying private LoRa networks, the hardware and software design are proposed in details. LoRa Gateway establishes the connection between massive LoRa Nodes and LoRa network server, so its function and performance are crucial. The paper focuses on the design and implementation of LoRa Gateway, which is able to support multi-mode communication and meets the needs of outdoor deployment. Moreover, an improved LoRa network server architecture is proposed. The architecture follows the principles of microservice, allowing for maximum control and flexibility. As a result, the LoRa network server is divided into four modularized and low-coupling modules, *i.e.*, Connector, Central Server, Join Server and Network Controller. They are responsible for different functions such as protocol analysis or node activation individually. Various advanced techniques are used in the implementation of LoRa network server to guarantee high performance and scalability. Modular design allows processing modules to be deployed in separated processes or servers and help users flexibly extend custom functionality in the corresponding module.

Aiming at providing insights about the design and implementation of LoRa system, the main contributions of this paper are summarized as follows:

1) The design and implementation of LoRa Gateway are proposed as a feasible solution.
2) We propose an improved LoRa network server architecture and implement it with an open-source project on GitHub[1].
3) Based on the proposed hardware platform and software architecture, a LoRa prototype system is deployed in urban environments and several experiments are conducted to evaluate the performance.

The rest of this paper is organized as follows. Section II presents some typical application scenarios of LoRa networks. Section III describes LoRa network architecture. Section IV gives the details on hardware design and implementation of LoRa Gateway. Section V proposes the improved architecture of LoRa network server. Section VI demonstrates and analyses the experimental results. Finally, section VII draws the conclusions.

[1]https://github.com/xisiot/lora-system

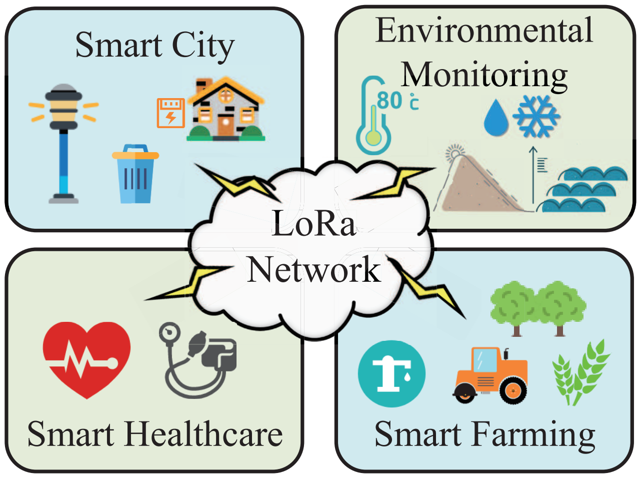

FIGURE 1: Illustration of typical application scenarios of LoRa networks.

II. TYPICAL APPLICATIONS OF LORA NETWORKS

With the communication range of over ten kilometers and long battery lifetime, LoRa network is promising for the long range, low power and low cost IoT applications. However, in order to achieve these excellent characteristics, LoRa technology comes at the expense of low data rate [15]. In this section, several typical applications and scenarios of LoRa network are discussed.

A. SMART CITY

LoRa network provides a reliable and feasible solution for various IoT applications in smart city, such as smart metering and waste management. More intelligently, these LoRa Nodes can automatically report sensing data and be controlled remotely through LoRa network. By connecting city services, these applications offer the opportunities to improve the efficiency of municipal operations and reduce maintenance costs.

B. ENVIRONMENTAL MONITORING

Environmental monitoring is crucial for our society to provide real-time environmental indicators, such as air quality, temperature and humidity, and alert to critical situations, such as natural disasters and environment pollution. Taking into account both financial and energy requirements, it is suitable to implement a network with LoRa technology across large geographical areas. The low-power LoRa Nodes with various sensors are able to interact with the environment and report data opportunely to detect issues before becoming crises.

C. SMART HEALTHCARE

Using information and communication technology in healthcare can lead to better treatment and disease surveillance for

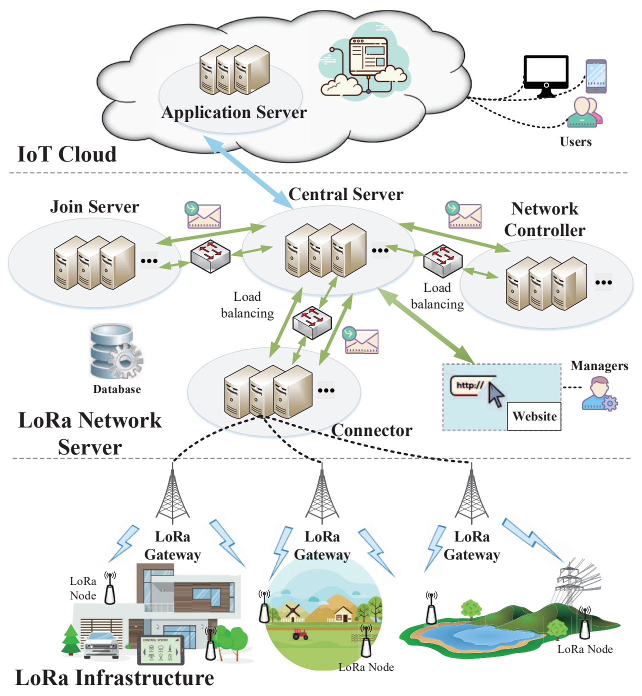

FIGURE 2: Illustration of LoRa network architecture.

patients. However, the vast cost burden is a challenge for the healthcare industry. Low cost and reliable performance of LoRa network make it suitable for typical smart healthcare applications. Various biological information is collected by special on-body sensors and can be checked in time. All abnormal indicators are transmitted to health service providers or health professionals through LoRa network for the early detection and prevention of diseases.

D. SMART FARMING

Smart farming is the application of modern technology into agriculture, aiming at improving the quantity and quality of agricultural products while reducing environmental impact and preserving resources. LoRa-based smart farming use cases, e.g., soil moisture monitoring and autonomous irrigation, have a great potential to deliver a more sustainable agricultural production. Long-range, low-cost and low-power features of LoRa technology enable the use of sensors to transfer information from the farm to the cloud where it can make more efficient operations and management.

III. LORA NETWORK ARCHITECTURE

As illustrated in Fig.2, a three-tier hierarchical LoRa network architecture is proposed. The functionalities of each component is described in this section.

A. LORA NODE AND GATEWAY

LoRa Node and Gateway are fundamental components of the entire LoRa networks since it can interact with the environment as well as forwarding data to LoRa network server for analysis. LoRa Node mainly includes sensors, actuators, battery, *etc*., and performs one or more tasks according to different application requirements. LoRa Gateway forwards all uplink radio packets to LoRa network server after adding metadata such as Signal-Noise-Ratio (SNR) and Received-Signal-Strength-Indicator (RSSI). Conversely, on the downlink, LoRa Gateway performs transmission requests coming from LoRa network server. Typically, it only transmits the received uplink/downlink packets as a relay without any interpretation of payloads. The hardware implementation details are described in the next section.

B. LORA NETWORK SERVER

LoRa network server receives packets forwarded from LoRa Gateway, and is responsible for packet processing and protocol analysis. Vast amount of functions need to be implemented on LoRa network server. For examples, checking the legality of end-devices, filtering duplicate uplink packets, performing network management mechanisms, scheduling acknowledgements and forwarding application layer data to the application server, *etc*. A well-designed architecture of LoRa network server is important for efficient processing and convenient management. Especially, the flexibility allows users to build various applications on existing LoRa hardware devices and the scalability can meet the needs of supporting massive LoRa Nodes. In addition, a management framework is necessary to help users to register, manage and monitor their LoRa devices including nodes and gateways. Aiming at providing insight about the improvement of LoRa network server, the proposed design and implementation of LoRa network server are presented in Section V.

C. APPLICATION SERVER IN IOT CLOUD

LoRa Application Server is responsible for encryption, decryption and processing of application layer payloads. It can support various applications with different encryption or encoding methods, such as Protocol Buffer, to ensure data security and improve transmission efficiency. IoT cloud provides essential services through Application Program Interfaces (APIs) and then applications are provided to users. LoRa Application Server acts as a bridge between IoT cloud and LoRa network server so that users can control LoRa Nodes and enjoy applications of LoRa network through web browsers or smart phones anywhere. A detailed explanation of IoT cloud can be obtained in [16].

IV. HARDWARE DESIGN AND IMPLEMENTATION

The hardware of LoRa system includes LoRa Node and LoRa Gateway. The design of LoRa Node is dependent on its specific application requirements and one typical example can be found in our published work [17]. In this paper, we only focus on the design and implementation of LoRa Gateway, which acts as a relay between LoRa Node and LoRa network server. Due to LoRa technology's long range capability, the LoRa Gateway is expected to cover dozens of square kilometers in the given environment. Therefore, it is necessary for LoRa Gateway to support two-way multi-

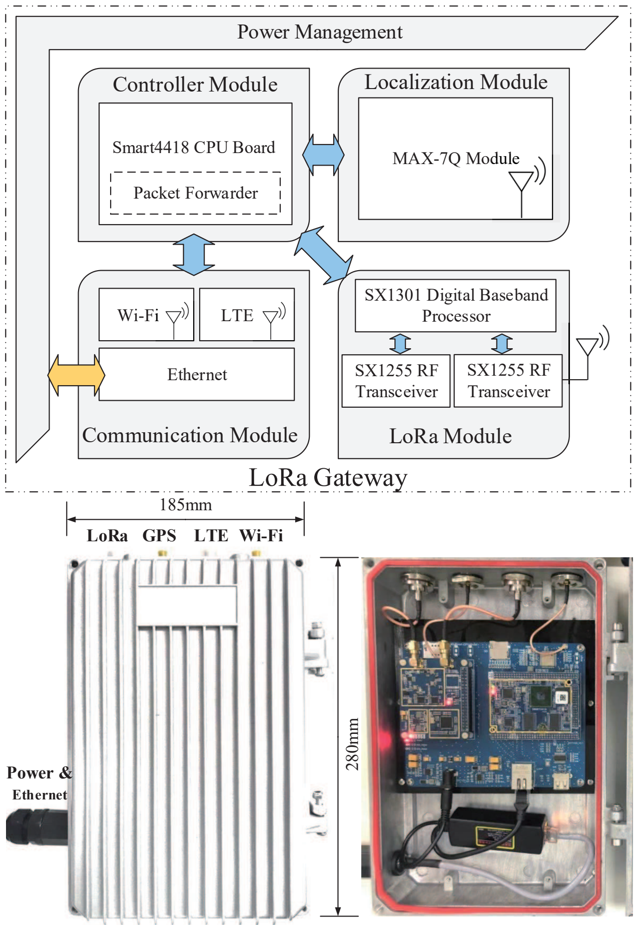

FIGURE 3: Illustration of LoRa Gateway and its modules.

channel communication based LoRa technology and have high signal receiving sensibility. Rich interfaces and high processing performance are the guarantee of scalability and large-scale access. In addition, LoRa Gateway should have good water-proof ability to support outdoor deployment.

As shown in Fig.3, the LoRa Gateway consists of five functional modules, *i.e.*, controller module, localization module, power management module, LoRa module and communication module. Its degrees of protection provided by enclosures can meet the requirements of IP65 level. The implementation details are given as follows, *i.e.*,

A. CONTROLLER MODULE

All control functions are implemented in controller module. The Smart4418 board is chosen due to its good performance, which uses a quad core Cortex A9 Computer Processing Unit (CPU) with dynamic frequency scaling up to 1.4 GHz and has 1 GB Random Access Memory (RAM) and 8 GB Embedded Multi Media Card (eMMC). Moreover, the board offers rich and advanced communication interfaces, such as up to 24 General-Purpose Input/Output (GPIO), 2 Serial Peripheral Interface (SPI) and Gbps Ethernet, *etc*. These features make Smart4418 suitable for implementation of LoRa Gateway.

In order to build a friendly runtime environment for software, the Debian system based on the Linux kernel is used in controller module. The packet forwarder program processes the communication protocol between LoRa Gateway and LoRa network server [18]. Based on this program, LoRa Gateway has the ability to connect with the server using specific packets through a User Datagram Protocol (UDP) link.

B. LORA MODULE

In order to achieve two-way multi-channel communication and excellent receiver sensitivity, the LoRa communication module consists of two SX1255 transceivers and one SX1301 digital baseband processor. SX1255 is an integrated RF transceiver and is designed to operate over the 400-510 MHz frequency bands. After SX1255 transceivers capture the RF signal, the data is concentrated into the SX1301 baseband processor via SPI. SX1301 contains 8 programmable parallel demodulation paths for LoRa channel and is able to demodulate simultaneously up to 8 packets. In addition, data-rate adaptation (ADR) scheme is supported by our designed LoRa Gateway. LoRa Node is able to transmit the packets with different data rate on a given channel as required.

C. COMMUNICATION MODULE

The LoRa Gateway has the other communication modules including Wi-Fi, LTE and Ethernet. Depending on the deployment environment, the reliable connection is able to be established from gateway to LoRa network server through one of communication mode.

D. LOCALIZATION MODULE

Localization module uses the U-blox's MAX-7Q for localization and time synchronization. The module can provide the position accuracy of less than 5 meters with low power and short search time from Global Positioning System (GPS). The GPS related information is transmitted to controller module through the serial port and then is parsed according to standard protocols.

E. POWER MANAGEMENT MODULE

Power Over Ethernet (POE) is adopted to simplify the deployment of LoRa Gateway. The power management module uses TPS54332D to offer 3.3 V and 5 V voltages for different components.

V. LORA NETWORK SERVER DESIGN AND IMPLEMENTATION

Aiming at providing an open flexible solution for both allowing users to conveniently deploying the LoRa network and supporting massive LoRa Nodes, we propose the improved architecture of LoRa network server and discuss its details as well as the implementation in this section.

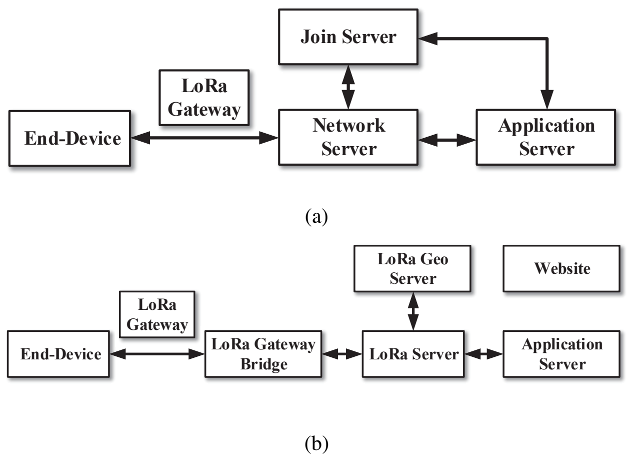

FIGURE 4: (a) The LoRa network architecture given by LoRaWAN. (b) The LoRa network architecture proposed by CableLabs [14].

A. RELATED WORKS ON ARCHITECTURE DESIGN

In the paper, we describe two typical architecture designs for comparison, i.e., the LoRa network architecture given by LoRaWAN and proposed by CableLabs [14], which are shown in Fig.4 (a) and (b) respectively.

In the reference architecture given by LoRaWAN, Network Server is the center of star topology and is expected to implement all the main functions. However, it may become a potential bottleneck because integrating excessive functionality requires more capabilities [19]. It also lacks of expansibility and cannot resist single point of failure. In addition, LoRaWAN only introduces the typical architecture without giving any concrete implementation. Different realizations may have a significant impact on the performance of LoRa network server.

As shown in Fig.4 (b), the architecture proposed by CableLabs separates two modules from LoRa Server, *i.e.*, LoRa Gateway Bridge and LoRa Geo Server [14]. They are designed to establish communication with LoRa Gateway and provide geolocation services respectively. Registration-related features such as handling join requests are integrated into LoRa Server. Compared with our proposed architecture, there are two main differences in the design concept. Firstly, the interaction between modules is implemented by gRPC while the messaging system is used in our proposed solution. The advantages of using the messaging system are elaborated later. On the other hand, similar to the architecture proposed by LoRaWAN, the centralized architecture leads to excessive functionality concentrated on one module, *i.e.*, LoRa Server. We propose a LoRa network server architecture that separates functionality into four flexible modules to attenuate the negative impact of centralization.

B. DESIGN OF LORA NETWORK SERVER

As illustrated in Fig.5, the improved architecture of LoRa network server is divided into four modules, *i.e.*, , Connector, Central Server, Join Server and Network Controller. It is

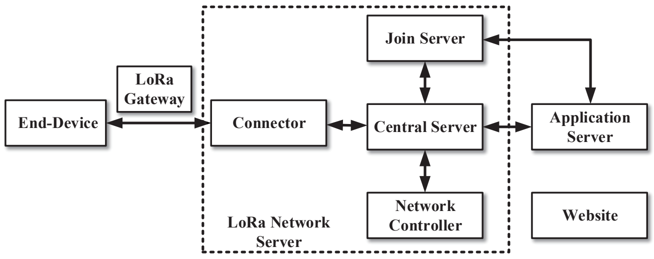

FIGURE 5: The improved architecture of LoRa network server.

worth mentioning that Connector performs packet parsing and discards illegal packets directly, which both improves security and reduces the occupation of computing resources. Network Controller is also separated from the traditional architecture in order to flexibly implement and perform network management schemes, such as Adaptive Data Rate (ADR) scheme.

Modularizing and splitting main functions into different modules can distribute the computational load as well as reducing performance requirements of the computer server, such as CPU and memory. The division of functions is in line with the principle of microservices in order to ensure low coupling between modules. Therefore, these modules can be deployed flexibly in separated processes or servers, allowing for agile deployment. All the modules in LoRa network server also can be quickly deployed by Docker containerization [20]. We package all the necessary components into a completed file system including project codes, databases and libraries. The method simplifies deployment steps as well as guaranteeing that each module runs correctly regardless of its environment. The key points in the architecture are elaborated as follows.

1) Interaction Between Modules

The interaction between modules is critical for the overall performance of LoRa network server. The publish-subscribe messaging system, *i.e.*, Apache Kafka, is used to establish reliable communication between models. As shown in Fig.6, the topic is a name of streaming data pipeline in messaging

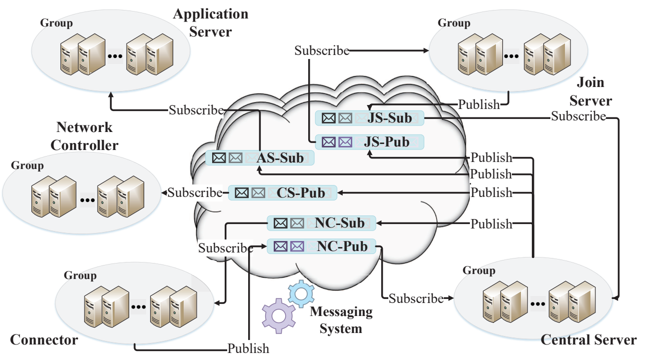

FIGURE 6: The design of interaction between modules.

system and different topics are designed for each module according to requirements. Each module subscribes one or more topics to fetch the corresponding data as a consumer and publishes messages to specific topics as a producer. The streaming data pipelines are acting as an abstract layer between consumer and producer. As long as the message format is specified, producers can be "ignorant" of the consumers. Moreover, the messaging system is able to provide both ordering guarantees and reliable asynchronous processing to address high concurrency challenges.

2) Load balancing

Load balancing is an essential design to handle a massive number of concurrent packets. In the messaging system, the consumer group has the concept of clustering the same modules for load balancing and fail-over. It is worth mentioning that each message published from the producer is assigned to one consumer in the subscribing consumer group. The scheduler operates the round-robin scheduling policy for all available consumers. Load balancing improves overall throughput and avoids bottlenecks caused by excessive load on a single processing module. In addition, load balancing also enhances the fault tolerance of LoRa network server. When a given single module fails, the data which needs to be processed can be immediately switched to other parallel modules without service interruption.

3) Database

In order to meet the storage requirements of heterogeneous data, Structured Query Language (SQL) and NoSQL databases are selectively used in LoRa network server. The persistent data with definite relationships, such as client intrinsic information, are stored in MySQL database. However, the main bottleneck of SQL database is the inefficiency in handling high concurrent read and write operations [21]. Consequently, NoSQL databases are able to provide real-time and high-efficiency services for data storage. Therefore, all session-related and non-persistent data, such as logs and duplicate packets, is stored by NoSQL databases. In addition, an efficient caching mechanism is necessary to ensure low latency and high performance when dealing with large amounts of concurrent packets.

C. IMPLEMENTATION OF LORA NETWORK SERVER

The implementation of LoRa network server are described with the functionality of each module. Aim at efficiently processing large amounts of packets, different types of content are assigned to different modules for processing. These modules are built on microservices principles with the goal of decoupling their duties and achieving distributed deployment. Moreover, a website project is developed as a management framework, which allows operators to register devices, monitor devices and retrieve historical data, *etc*.

1) Connector

Connector is the bridge between LoRa Gateway and Central Server, which provides the services for parsing and packaging the payload based on LoRaWAN specification [22]. LoRa Gateway connects with a given Connector through an IP back-bone and uses UDP for packet transmission. Firstly, Connector performs legality verification on LoRa Nodes and Gateway to avoid unauthorized access. For example, if LoRa Gateway has been legally registered, Connector immediately replies an acknowledgment to indicates that the packet is received. Otherwise, the packet is discarded directly. Moreover, parsing and packaging of a packet require many necessary database operations. Separating the function from traditional network server performs well to spread the processing pressure and improve the utilization of all accessible resources. On the other hand, packets associated with LoRa node activation process are ignored and forwarded directly to Central Server.

2) Central Server

Central Server is responsible for data management and service scheduling. Different modules are scheduled by Central Server according to the requirements of data processing. Depending on the type of uplink packet, the information in packet is separated into the specific formats in order to facilitate subsequent processing. Then, the original application data is fed into Application Server, the MAC commands are sent to Network Controller and the join packets are forwarded to Join Server. In addition, the transmitted data is collected and stored by Central Server. Managers can use web browsers to check up the uplink/downlink packets and monitor the running states of LoRa node and Gateway in real time.

Since LoRa Nodes are not associated with a specific LoRa Gateway, the single packet may be received by multiple gateways simultaneously. To avoid the waste of radio resources due to redundancy, Central Server is essential for filtering duplicate packets. Only one of duplicate packets is fed into the subsequent processing module. Moreover, the responsibility of Central Server includes scheduling of downlink packets. The transmission information such as RSSI and SNR attached in the duplicate packets is not discarded and then one optimal gateway is selected to send the downlink packet through exploiting this information. Finally, Central Server determines the downlink parameters according to related configurations and reads the downlink payload from two data queues which are responsible for application data and MAC commands.

3) Join Server

The services provided by Join Server include handling activation requests and generating session keys. There are two methods to activate LoRa Nodes, *i.e.*, Over-The-Air Activation (OTAA) and Activation By Personalization (ABP). For OTAA method, LoRa Node must follow a join procedure consisting of the join request and the join accept exchange. To secure the activation process, the join request packet is

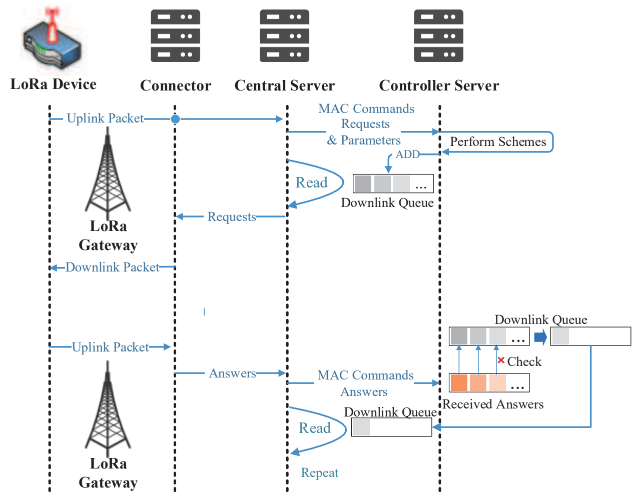

FIGURE 7: Illustration of the workflow of Network Controller.

only parsed by Join Server and then the session keys are derived from both the pre-shared key and the transmitted random number. Finally, the join accept packet is encapsulated and encrypted by the corresponding key in Join Server. Furthermore, LoRa Nodes using ABP method need to be registered on the website through specific management interfaces.

4) Network Controller

The functions of Network Controller focus on processing and managing MAC commands. MAC commands are exchanged between Network Controller and LoRa Nodes in order to modify associated configurations or adjust transmission parameters [22]. Network Controller is stripped from classic LoRa network server, which flexibly performs Mac commands and new network management schemes. The well-designed schemes such as Adaptive Data Rate (ADR) are implemented since they can improve both reliability and capability of LoRa networks [23]. Fig.7 shows the workflow of Network Controller. Take the implementation of ADR scheme as an example, the transmission parameters of the uplink packets are conveyed from Central Server. Then Network Controller performs the ADR scheme and determines whether to adjust either the data rate or transmission power. Finally, the MAC commands are added to the downlink queue and sent by the downlink packet. When receiving the acknowledgement packet, the successful request commands are removed from the queue. Otherwise, the failed commands are reserved for the next transmission.

VI. EXPERIMENTAL RESULTS AND ANALYSIS

In order to evaluate the performance of proposed LoRa network, a prototype LoRa network has been implemented and deployed in typical urban environments. Not only the coverage but also the network server performance of LoRa system are evaluated in this section.

TABLE 1: LoRa Node Parameters

Parameters	**Value**
Carrier frequency	433 MHz
Preamble	8 $Symbol$
Transmit power	20 dBm
Coding rate	4/5
Spreading factor	12
Bandwidth	125 kHz

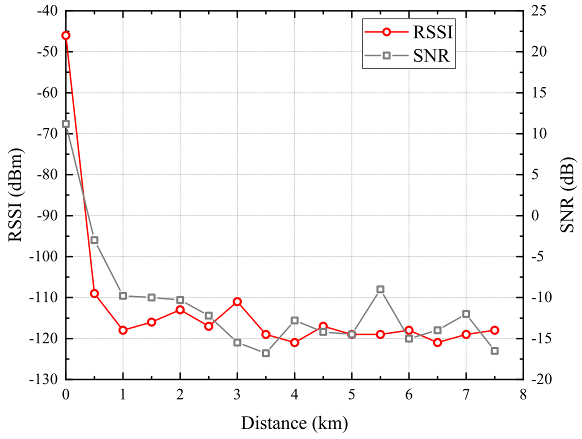

FIGURE 8: Coverage performance of LoRa transmission.

A. COVERAGE PERFORMANCE AND ANALYSIS

The LoRa Gateway with whip antenna is vertically installed on the roof of a 15-floor building at the center of our campus. The omnidirectional antenna has a gain of 10 dBi and an operating frequency range from 400 MHz to 470 MHz. Then, the packets are collected through LoRa transmission and then forwarded to LoRa network server by LTE [17]. Both the LoRa Gateway and Nodes operate at the frequency band of 433 MHz. In order to evaluate the maximum distance of the LoRa transmission, LoRa Nodes adopt the maximum spreading factor, *i.e.*, SF=12, to transmit the packets. The main parameters in field trials are given in Table 1.

In the experiments, several LoRa Nodes continuously send uplink packets and gradually move away from the location of LoRa Gateway. The received signal quality parameters are measured about every 0.5 km. As shown in Fig.8, the RSSI and SNR decrease sharply within 1 km, while vary slowly with the increase of distance. It can be seen that the maximum distance that LoRa Gateway can receive the uplink packets is about 7.5 km, while the minimal RSSI and SNR is -118 dBm and -16.5 dB respectively.

B. LORA NETWORK SERVER PERFORMANCE AND ANALYSIS

In order to evaluate the network performance of LoRa system, the proposed modules and the stress testing program are deployed on two separated VMs. The main parameters of the VMs are shown in Table 2. The testing program is developed based on Locust, which can emulate thousands of concurrent users on a single VM [24]. In our experi-

TABLE 2: Virtual Machine Parameters

Parameters	Server VM	Test VM
CPU	Intel(R) Xeon(R) CPU E5-2680	Intel(R) Xeon(R) CPU E5-2609 v4
RAM	8 *GB*	16 *GB*
Number of logical CPU	5	8
Operating System	Ubuntu 14.04.3 LTS	Ubuntu 16.04.3 LTS

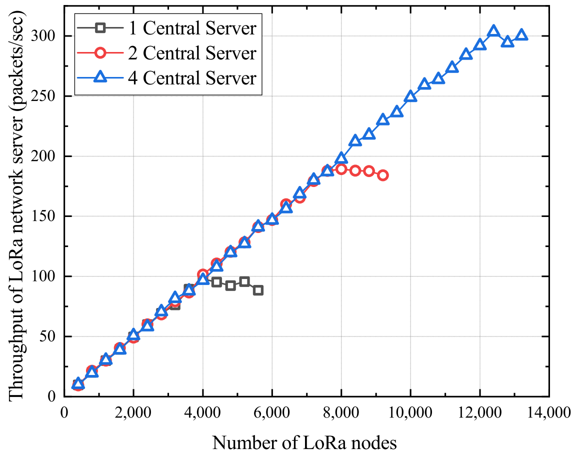

FIGURE 9: Throughput of LoRa network server.

ments, each LoRa node is treated as a user and generates an uplink application packet about every 40 seconds. We assume that all the packets are successfully received by LoRa Gateway since LoRa Gateway has multi-channel communication capability and powerful processing performance. The testing program is designed to directly send/receive the uplink/downlink packets by UDP to/from the network server [18]. The number of LoRa Nodes is set from 400 to 14,000 as required. Each uplink packet should be replied and the response packets are received by LoRa Nodes if the network server handles the packets successfully. Otherwise, if no response is received within 5 seconds, the processing of uplink packet is regarded as a failure. When the number of LoRa Nodes increases, the throughput under the given situation is measured. Meanwhile, the median response time and average CPU utilization of Connector and Central Server are also used as reference indicators. Our experiments are mainly focused on Connector and Central Server since both of them have significant impacts on the quality of services under consideration.

As shown in Fig.9, the throughput of LoRa network server is linear with the increasing number of LoRa Nodes in the region of small number since LoRa network server can successfully process all packets. However, the throughput becomes floor when the number of LoRa node reaches a specific limit, e.g., 4000, 7,600, and 12,400 for 1, 2 and 4 Central Server, respectively. Meanwhile, the maximum throughput is about 98, 190 and 300 packets per second. It is because that too many packages cause the network server

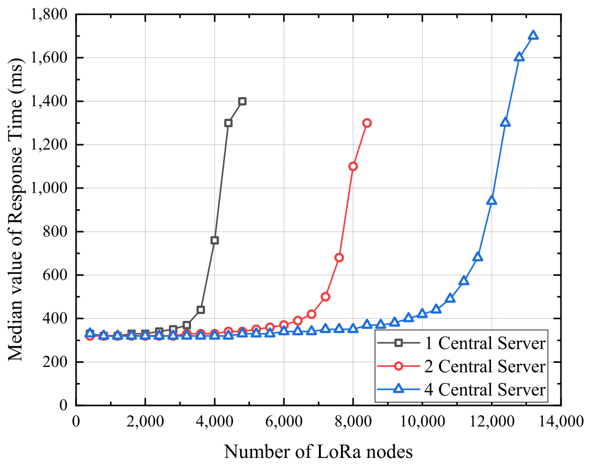

FIGURE 10: The median value of response time.

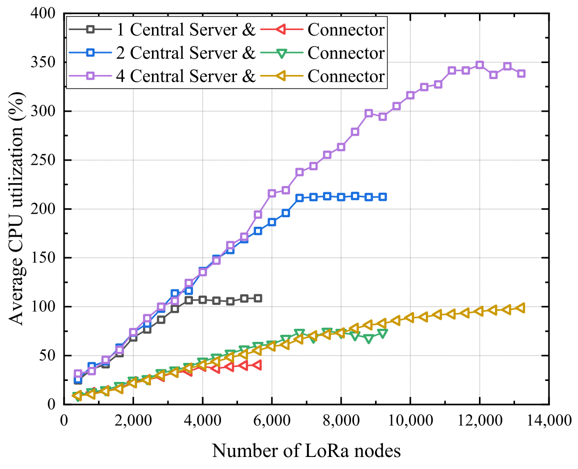

FIGURE 11: Average CPU utilization of Connector and Central Server.

unable to successfully process packets in time. Figure 10 shows that the median value of response time changes slowly when the number of Nodes is less than the limit but increases sharply when the number becomes too larger.

Figure 11 shows the average CPU utilization of Central Server and Connector for different deployment number. It can be seen that, the CPU utilization of Central Server can slightly exceed 100% and 200% with 1 or 2 Central Server and the curves no longer rise with the increase of LoRa Nodes. In addition, when the CPU utilization of Connector reaches 100%, the CPU utilization of Central Server also becomes floor. It is because these modules are a single-threaded program developed by Node.js. One Connector or Central Server can only occupy one logical CPU. When any of the logic CPU is fully occupied no matter on which module is running, LoRa network server cannot accommodate more LoRa Nodes. On the other hand, Central Server needs more computing resources than Connector so that it is more likely to become a bottleneck. Luckily, our proposed architecture supports deploy multiple modules distributedly and achieves load balancing across them.

VII. CONCLUSION

Due to wide area coverage and low power consumption of LoRa network, various applications provided by LoRa have been emerging in the IoT market. In this paper, aiming at providing a flexible and completed solution for building a private LoRa network, the design and implementation of LoRa network have been proposed including hardware and software. Moreover, the open source project is available on GitHub. The field trails demonstrate that the maximum transmission distance based on our hardware design is about 7.5 *km* in urban environments. Furthermore, we also show that LoRa system under the improved architecture can support more than 10,000 LoRa Nodes with well-deployed computing resources. It is expected that this open LoRa network can provide the flexibility and feasibility for both academic and industry to deploy their new LPWA applications in the near future.

ACKNOWLEDGMENT

The work was supported by the National Natural Science Foundation of China (NSFC) under the Grant Number 61671089.

REFERENCES

[1] *LoRa Alliance*, "LoRa Alliance 2017 end of year report." Available: https://lora-alliance.org/sites/default/files/2018-04/LoRa-Alliance-Annual-Report.pdf. [Online; accessed Mar. 15, 2019].

[2] *LoRa Alliance*, "LoRaWAN What is it? A technical overview of LoRa and LoRaWAN." Available: https://lora-alliance.org/sites/default/files/2018-04/what-is-lorawan.pdf. [Online; accessed Mar. 15, 2019].

[3] X. Xiong, K. Zheng, R. Xu, W. Xiang, and P. Chatzimisios, "Low power wide area machine-to-machine networks: key techniques and prototype," IEEE Communications Magazine, pp. 64–71, Sep. 2015.

[4] J. de Carvalho Silva, J. J. P. C. Rodrigues, A. M. Alberti, P. Solic, and A. L. L. Aquino, "Lorawan — a low power wan protocol for internet of things: A review and opportunities," in International Multidisciplinary Conference on Computer and Energy Science, pp. 1–6, July 2017.

[5] R. Sinha, Y. Wei, and S.-H. Hwang, "A survey on lpwa technology: Lora and nb-iot," ICT Express, pp. 14–21, 2017.

[6] M. C. Bor, U. Roedig, T. Voigt, and J. M. Alonso, "Do lora low-power wide-area networks scale?," in Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, pp. 59–67, 2016.

[7] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, "Understanding the limits of lorawan," IEEE Communications Magazine, vol. 55, pp. 34–40, Sep. 2017.

[8] D. Magrin, M. Centenaro, and L. Vangelista, "Performance evaluation of lora networks in a smart city scenario," in IEEE International Conference on Communications, pp. 1–7, May 2017.

[9] J. L. Pérez and D. Carrera, "Performance characterization of the serviotic api: An iot-as-a-service data management platform," in IEEE First International Conference on Big Data Computing Service and Applications, pp. 62–71, Mar. 2015.

[10] G. Merlino, D. Bruneo, S. Distefano, F. Longo, and A. Puliafito, "Enabling mechanisms for cloud-based network virtualization in iot," in IEEE 2nd World Forum on Internet of Things, pp. 268–273, Dec. 2015.

[11] K. Zheng, H. Meng, P. Chatzimisios, L. Lei, and X. Shen, "An smdp-based resource allocation in vehicular cloud computing systems," IEEE Transactions on Industrial Electronics, vol. 62, pp. 7920–7928, Dec. 2015.

[12] J. So, D. Kim, H. Kim, H. Lee, and S. Park, "Loracloud: Lora platform on openstack," in 2016 IEEE NetSoft Conference and Workshops, pp. 431–434, June 2016.

[13] *The Things Network*. Available: https://www.thethingsnetwork.org. [Online; accessed Mar. 15, 2019].

[14] *LoRa Server, Open-source LoRaWAN Network-server*. Available: https://www.loraserver.io/. [Online; accessed Mar. 15, 2019].

[15] U. Raza, P. Kulkarni, and M. Sooriyabandara, "Low power wide area networks: An overview," vol. 19, no. 2, pp. 855–873.

[16] L. Hou, S. Zhao, X. Xiong, K. Zheng, P. Chatzimisios, M. S. Hossain, and W. Xiang, "Internet of things cloud: Architecture and implementation," IEEE Communications Magazine, vol. 54, pp. 32–39, Dec. 2016.

[17] W. Zhao, S. Lin, J. Han, R. Xu, and L. Hou, "Design and implementation of smart irrigation system based on lora," in IEEE Globecom Workshops (GC Wkshps), pp. 1–6, Dec. 2017.

[18] *Semtech*, "Lora network packet forwarder project." Available: https://github.com/Lora-net/packet_forwarder. [Online; accessed Mar. 15, 2019].

[19] K. Zheng, S. Ou, J. Alonso-Zarate, M. Dohler, F. Liu, and H. Zhu, "Challenges of massive access in highly dense lte-advanced networks with machine-to-machine communications," IEEE Wireless Communications, vol. 21, pp. 12–18, June 2014.

[20] *Docker*, "What is locust?." Available: https://docs.docker.com. [Online; accessed Mar. 15, 2019].

[21] Y. Li and S. Manoharan, "A performance comparison of sql and nosql databases," in IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, pp. 15–19, Aug. 2013.

[22] *LoRa Alliance*, "Lorawan specification (v1.1)." Available: https://lora-alliance.org/resource-hub/lorawantm-specification-v11. [Online; accessed Mar. 15, 2019].

[23] F. Liu, K. Zheng, W. Xiang, and H. Zhao, "Design and performance analysis of an energy-efficient uplink carrier aggregation scheme," IEEE Journal on Selected Areas in Communications, vol. 32, pp. 197–207, Feb. 2014.

[24] *Locust documentation*, "What is locust?." Available: https://docs.locust.io/en/stable/what-is-locust.html. [Online; accessed Mar. 15, 2019].