

## Research article

# DSME-FOTA: Firmware over-the-air update framework for IEEE 802.15.4 DSME MAC to enable large-scale multi-hop industrial IoT networks

Jabeom Gu, Seung-Sik Lee, Hoyong Kang\*

*Electronics and Telecommunications Research Institute, 218 Gajeong-ro, Daejeon, 34129, Republic of Korea*

## ARTICLE INFO

## Keywords:

Firmware over-the-air  
DSME  
DSME-FOTA  
Internet of Things  
Industrial IoT  
Multi-hop network  
IEEE 802.15.4  
Wireless sensor network  
LPWA  
LoRa

## ABSTRACT

The Internet of Things (IoT) is rapidly expanding in applications ranging from smart homes and smart cities to various other domains, including agriculture, transportation, industrial automation, and environmental monitoring. Regardless of the application domain, firmware update over-the-air (FOTA) capability is critical for any IoT network, and its importance is particularly evident in the field of industrial IoT. This paper presents a comprehensive solution designed for large-scale industrial IoT networks, with a special focus on Deterministic Synchronous Multi-channel Extension (DSME)-based multi-hop networks, called DSME-FOTA. The paper provides a detailed FOTA protocol design, a numerical analysis, and proof-of-concept experiments. The analytical and experimental results show that the total firmware update time of DSME-FOTA exhibits a first-order dependence on both the number of hops and the firmware binary size.

## 1. Introduction

The Internet of things (IoT) is gaining popularity in a variety of application scenarios, not only in small-scale implementations like smart homes [1], but also in large-scale use cases including smart buildings, cities [2], metering [3], vehicular tracking [4], and infrastructure or environment monitoring [5,6]. In this diverse IoT landscape, Low-Power Wide-Area Networks (LPWAN) technologies like LoRaWAN, Sigfox, and NB-IoT [7–9] are playing key role, enabling connectivity over long distances.

For industrial IoT (IIoT) applications [10], however, specific requirements demand robust and reliable communication protocols as well. Multi-hop networks, characterized by mesh-like structures with relatively short radio distances compared to conventional LPWAN, are particularly effective for real-time monitoring in complex industrial environments such as chemical plants, power plants, large vessels, or wind farm [11–16]. These settings often feature structural complexities, heavy machinery, metallic infrastructure, and extensive piping systems, necessitating a substantial number of IoT nodes for comprehensive monitoring and control. The LPWAN technology may encounter coverage limitations in such conditions [17]. Multi-hop communication allows IoT devices to relay data through intermediate nodes, extending the communication range and overcoming coverage limitations posed by complex industrial infrastructures.

The IEEE 802.15.4 Deterministic Synchronous Multi-channel Extension (DSME) [18–20], introduced in the IEEE 802.15.4e standard in 2012 [21], addresses the critical needs of industrial automation and control systems by prioritizing predictable latency and reliable data transfer. DSME uniquely combines contention-based and contention-free channel access methods, integrating advanced features like channel adaptation (CA) and channel hopping (CH) to enhance communication robustness and minimize

\* Corresponding author.

E-mail addresses: [gjb@etri.re.kr](mailto:gjb@etri.re.kr) (J. Gu), [rfslee@etri.re.kr](mailto:rfslee@etri.re.kr) (S.-S. Lee), [hoyong.kang@etri.re.kr](mailto:hoyong.kang@etri.re.kr) (H. Kang).

interference and transmission failures. Furthermore, DSME extends and adapts the Guaranteed Time Slot (GTS) concept from the IEEE 802.15.4 standard, specifically optimizing it for multi-hop mesh networks prevalent in industrial settings [22]. DSME's design is strategically tailored to meet the stringent requirements of industrial IoT use cases, ensuring predictable latency, scalability, and reliability in challenging operational environments [23,24].

However, for practical applicability in industry, the IEEE 802.15.4 DSME standard still needs to integrate an over-the-air firmware update scheme. Once a network is deployed, there is often a need to update each device's firmware over the network's lifespan, either to introduce new features or to address security vulnerabilities [25–27]. Given that IoT network nodes can be dispersed across extensive operational areas, individually upgrading each device significantly increases management time and costs.

Updating Firmware Over-the-Air (FOTA) is a practical solution to overcome this issue, in which the IoT network itself is exploited to deliver a new firmware to each device, thus significantly enhancing scalability and operational efficiency.

This paper focuses on addressing the missing OTA firmware update capability within the IEEE 802.15.4 DSME standard, specifically tailored for multi-hop Industrial IoT (IIoT) deployments. By enabling OTA firmware updates in DSME-based multi-hop networks, we aim to enhance the robustness, scalability, and efficiency of IIoT deployments, particularly in challenging industrial environments. Operating primarily at a lower level within the network stack, it ensures efficient utilization of network resources. This protocol enables efficient and standardized firmware updates within the network, simplifying the process of deploying updates across multiple devices. It is carefully integrated with existing DSME protocols and network management functionalities, ensuring seamless compatibility and interoperability. By operating in the network stack, it eliminates the need for application-level protocol implementations, thereby simplifying network architecture and reducing overhead.

The main contributions of this paper are as follows:

- First, we provide a detailed design of the IoT network using the LoRa physical layer (PHY) and a proprietary implementation of the IEEE 802.15.4 Distributed Synchronous Multi-channel Extension (DSME) of the Medium Access Control (MAC) layer. The decision to use the LoRa PHY instead of the original IEEE 802.15.4 PHY was driven by LoRa's inherent advantages in handling interference within industrial environments. LoRa technology excels in mitigating interference caused by heavy machinery, metallic infrastructure, and other industrial equipment, making it well-suited for dispersed IoT nodes and addressing potential coverage challenges.
- Second, the DSME-FOTA framework is designed with a sequential cascade of master–slave operations, starting from the Access Point (AP) and extending through multiple routers, to ensure the successful distribution of new firmware binaries to all network devices.
- Third, we conduct a detailed numerical analysis of the overall performance, particularly in terms of the FOTA update time, providing quantitative insights into the effectiveness of the framework. The result also offers a hint for predicting the expected FOTA update time when applying DSME-FOTA to any given network configuration.
- Finally, we evaluate the implemented system in the laboratory through a series of proof-of-concept experiments and performance evaluations.

The remainder of this paper is structured as follows: Section 2 presents related work. Section 3 discusses the implementation of DSME and the formation of multi-hop networks. Section 4 presents the details of the DSME-FOTA framework. Section 5 provides a numerical analysis of the performance in terms of FOTA update time. To validate the approach, Section 6 presents experiments conducted in a laboratory testbed, providing practical insights into the real-world viability of the solution. Finally, in Section 7, we conclude the paper.

## 2. Related work

The extension of IoT capabilities into the industrial domain will enable the redesign and modularization of the factory floor [28–31]. Moreover, IoT technology is increasingly applied to areas demanding extreme reliability, real-time performance, and predictable latency [32–34].

Notable applications include large chemical plants [11], network infrastructure within shielded buildings of nuclear power plants (NPPs) [12,13], smart Electric Power and Energy Systems (EPEs) [12], and wind turbine [35]. The paper [11] presents a survey on IoT applications in large-scale petrochemical plants, focusing on middleware approaches that enable rapid deployment of wireless sensor networks and support crowdsensing-based services. Authors in [36] explore the application of IoT in wind farms to enhance operational efficiency, reduce costs, and contribute to global renewable energy goals.

In large-scale deployments like shipboard Wireless Sensor Networks (WSNs) [37], hierarchical zone-based networks have been proposed using IEEE 802.15.4 compliant hardware. Research on radio signal propagation within ships has been conducted, exploring both sub-1 GHz [14] and 2.4 GHz [15] bands. Additionally, studies have examined the suitability of industrial WSN devices for high-radiation environments [38].

However, regardless of the application domain, firmware update over-the-air (FOTA) capability is critical for any IoT network, and its importance is particularly evident in the field of industrial IoT. The FOTA in IoT networks has a similar concept as earlier works in WSNs [39,40]. Therefore, the main difficulties and design considerations encountered in updating IoT firmware can also be found in WSNs.

In [41], the difficulties encountered in IoT FOTA are discussed, namely: (a) Data size — the firmware binary size is typically larger than the regular sensor data, (b) Scalability — the firmware needs to be transported to thousands or more nodes, (c) Reliability — the firmware needs to be correctly received by all nodes regardless of transmission errors, temporary disconnections, etc., (d)

**Table 1**  
Comparison of Firmware Over-the-Air update schemes.

Reference	Topology	Objective	Method	Configuration	1-hop FOTA update time
[42]	Star	LoRaWAN OTA using Broadcast	Simulation	LoRaWAN, SF7/BW125 kHz	28 min
[43]	Multi-hop	Reduce packet transmissions	Implementation	Zigbee, 38.4 kbps	92 min
[48]	Star	IETF 9019 based OTA implementation	Implementation	WiFi, N/A	2.5 s
[49]	Star	NB-IoT OTA experiments	Implementation	NB-IoT, N/A	7.2 min
[45]	Mesh	Mesh network OTA Simulation	Simulation	N/A	166 min
Proposed	Multi-hop	OTA for DSME based multi-hop network	Implementation	DSME, SF5/BW500 kHz	7.4 min

Update time — should be minimized to reduce service interruption. In [42], the authors further discuss the design challenges in implementing FOTA for IoT networks: (a) low data rate transmission, (b) regulation allowing low duty cycle in the sub-1 GHz Industrial Scientific and Medical (ISM) bands (433/868/915 MHz). The design should also consider the scarce resources (computing, battery, storage, and radio transmission power) of the IoT device as well [43]. Arakadakis et al. [44] present recent advances in FOTA methods in IoT networks, and highlight challenges and limitations.

As these aspects are critical to success, most of the research has been carried out to mitigate these issues, and generally, the architecture and protocol show an absolute dependence on the underlying transport. Current IoT networks are being developed using several different communication technologies, broadly categorized as Low-Rate Wireless Personal Area Network (LR-WPAN) and Low Power Wide Area Network (LPWAN). IEEE 802.15.4 and Zigbee belong to the former, while LoRaWAN, Sigfox, and NB-IoT have become de facto standards for the latter.

Numerous studies have tackled FOTA approaches in IoT networks, ranging from solutions for low data rate transmissions to overcoming regulatory constraints in the sub-1 GHz ISM bands. The paper [45] presents simulation of number of messages and completion time for FOTA in mesh networks. Authors in [42] present design challenges in LoRaWAN and evaluate the performance of a recent change to the standard, which is proposed to enable efficient FOTA in LoRaWAN. The paper [46] investigates the feasibility of providing FOTA updates for emerging LPWAN technologies in terms of energy consumption. It compares IEEE-802.15.4g, LoRa, and SigFox for three different scenarios: full firmware update, application update, and network protocol update.

Recent Internet Standards activity includes IETF Request for Comment (RFC) 9019 [47], which defines a firmware update architecture specifically designed for Internet of Things (IoT) devices with resource constraints. Regarding this, the authors [48] propose a FOTA solution for constrained IoT devices using the RFC 9019, in which the experiment was carried out using the WiFi devices such as NodeMCU and Raspberry Pi.

The study in [27] not only tracked the latest advances in FOTA but also focused on security threats and challenges.

The study [49] showcases a firmware over-the-air (OTA) update approach for resource-constrained IoT devices, employing Narrowband IoT (NB-IoT) in licensed spectrum. NB-IoT can theoretically support over 50,000 devices per cell.

In [50], the authors present an over-the-air firmware update system based on a mesh network protocol called LightweightMesh. The LightweightMesh is designed to provide low-power mesh and route discovery & establishment for smart urban development. In [51], the authors propose an incremental FOTA which is designed to apply an energy harvesting environment monitoring system, so that the firmware is updated incrementally while the device is still running (no reboot is required after FOTA).

In summary, Table 1 presents an overview of existing works in the field, highlighting their respective 1-hop Firmware Over-the-Air (FOTA) update times in comparison to the proposed scheme. The majority of IoT networks explored in the literature utilize a star topology, with limited coverage of multi-hop architectures. Additionally, the predominant focus of these works is on application-level OTA implementations.

In this paper, we focus on addressing the critical gap in OTA capabilities within the DSME network, specifically tailored for large-scale multi-hop Industrial IoT (IIoT) deployments. Our approach involves utilizing the LoRa (Long Range) transmission technique as the Physical (PHY) layer in conjunction with the DSME MAC. This strategy is similar to the approach proposed in [52], where authors explored the use of DSME with LoRa to mitigate application limitations in LoRaWAN caused by various factors.

### 3. System design

In this section, we present the implementation of the DSME, where we build a versatile IoT network using the LoRa physical layer (PHY) and a proprietary implementation of the DSME MAC.

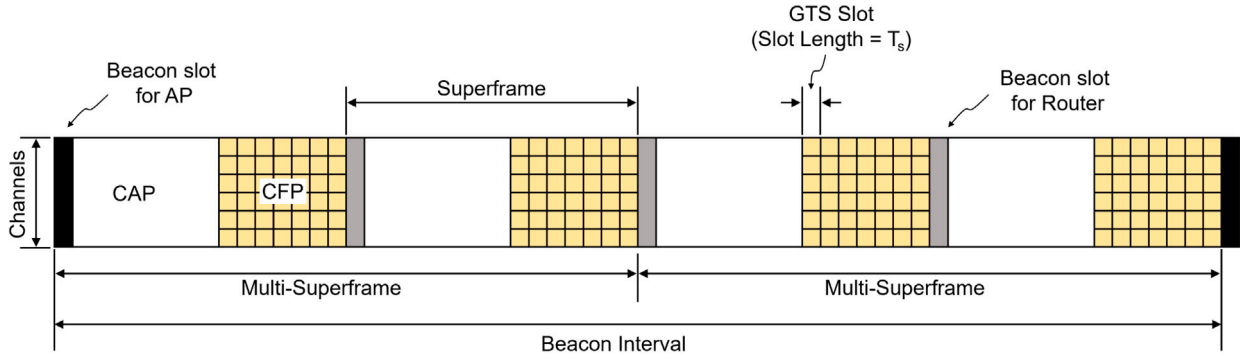


Fig. 1. Multi-superframe structure of the IEEE 802.15.4 DSME MAC.

### 3.1. Basics of the DSME

This section briefly introduces the IEEE 802.15.4 Deterministic Synchronous Multi-channel Extension (DSME). The DSME is a Medium Access control (MAC) protocol introduced in IEEE 802.15.4 [21]. It is intended for applications that need high reliability, bounded latency, and scalability.

The DSME uses the same definition of the superframe as IEEE 802.15.4, which originally consisted of three parts: a Beacon Slot, a Contention Access Period (CAP), and a Contention-Free Period (CFP). In addition, as illustrated in Fig. 1, the DSME also specifies a multi-superframe structure, meaning that multiple superframes exist within a single Beacon Interval.

The Beacon Slot is used by an Access Point (AP) or a router, a node that is allowed to broadcast Beacon packets, to transmit Beacons.

The CAP is utilized to schedule time slots for data transmission, exchange control messages between nodes, and more. As the name implies, a node that has obtained channel access through CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) contention has the right to transmit its data. Because more than one node can transmit at the same time, a collision occurs during this period.

During the CFP interval, each node transmits data in its pre-assigned time slot and channel. The CFP in the DSME MAC preserves the IEEE 802.15.4 superframe structure while ensuring the timely delivery of sensor data.

The time slots within this period are called Guaranteed Time Slots (GTSs) and are allocated to wireless nodes to guarantee transmission opportunities.

In DSME, each GTS transmission is designed to use a different frequency, enabling the RF wireless link to achieve a high level of reliability through the use of the channel hopping (CH) mechanism. In the implementation, the channel frequency accessed during each GTS slot is changed according to a predetermined channel hopping sequence. The CH sequence cycles back to the first channel when it reaches the end of the sequence.

Due to DSME MAC's support for multi-superframes, it can be easily extended to various types of multi-hop networks, including star, cluster-tree, partial mesh, and more.

The superframe structure in the DSME MAC is determined by the values of Beacon Interval (BI), superframe duration (SD), and multi-superframe duration (MD). The SD represents a superframe length, that is determined by the value known as the superframe order (SO).

The length of a Time Slot is defined as:

$$T_s = aBaseSlotDuration \times 2^{SO} \text{ [symbols]}, \quad (1)$$

where  $aBaseSlotDuration$  is fixed at 60 symbols, and there are 16 slots in each superframe (One for Beacon, 8 slots for CAP, and 7 GTS slots for CFP). Therefore the  $aBaseSuperframeDuration$  is 960 symbols. The resulting superframe duration is as follows:

$$SD = aBaseSuperframeDuration \times 2^{SO} \text{ [symbols]} \quad (2)$$

Because the SD is determined by the SO, the length of the CAP and CFP slots can be adjusted depending on the size of the sensor data.

The BI defines the interval at which Beacon packets are transmitted by the AP, providing information related to network configuration and synchronization. The MD determines the number of superframes within a BI, as changing the MO changes the number of GTS slots. The purpose of the MO is to address the issue that the duration of the GTS slot allocated to each node device may increase as the BI value increases, potentially leading to increased delay in the transmission of sensor information. In other words, each end device (ED) may have a shorter period for its assigned GTS slot when multiple multi-superframes are applied within a BI. Both BI and MD can also be determined using the values of the beacon order (BO) and the multi-superframe order (MO) values:

$$BI = aBaseSuperframeDuration \times 2^{BO} \text{ [symbols]} \quad (3)$$

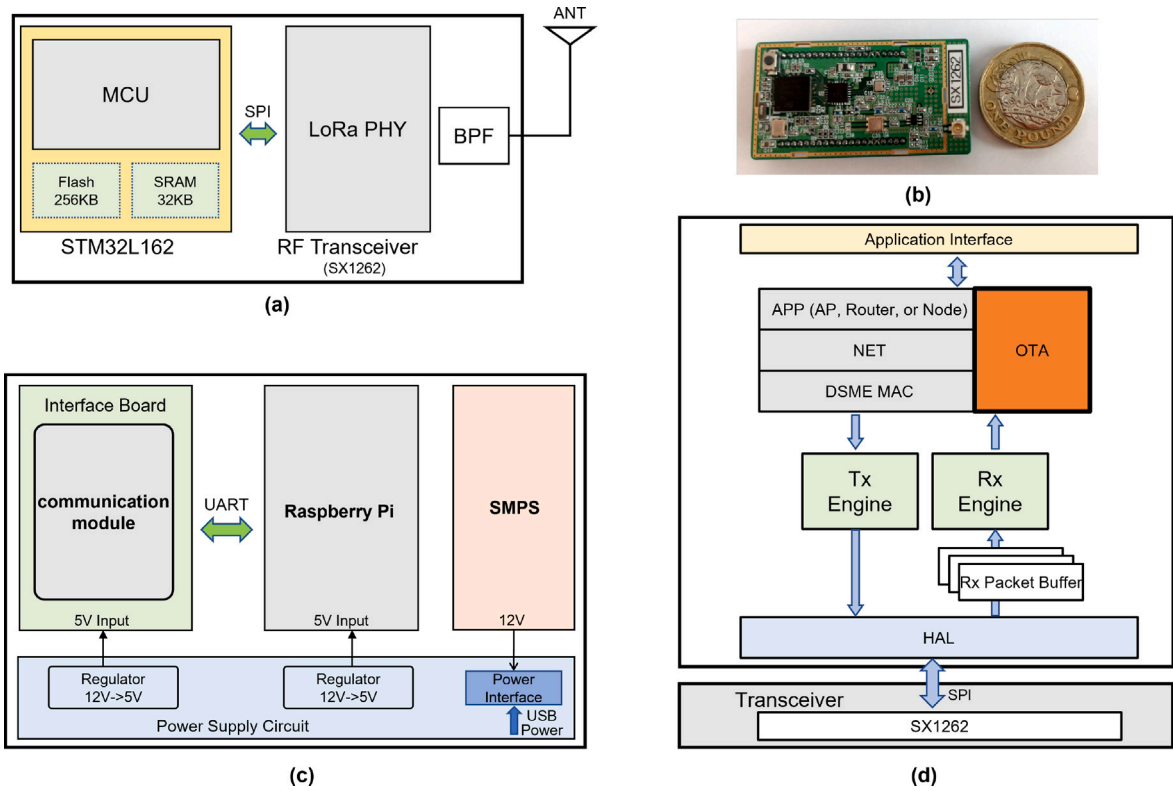


Fig. 2. Hardware and software implementation: (a) Hardware structure of the communication module, (b) Photograph of the communication module, (c) Access Point (AP) block diagram, (d) and Software structure of the communication module.

$$MD = aBaseSuperframeDuration \times 2^{MO} \text{ [symbols]} \tag{4}$$

The number of GTS slots within the Beacon Interval can be changed by adjusting the MO.

### 3.2. Hardware and software implementation

The detailed design of our implementation is illustrated in Fig. 2. This paper utilizes a Cortex-M3 based microcontroller (STM32L162) paired with a Semtech SX1262 LoRa transceiver, as depicted in Fig. 2(a). The SX1262 transceiver is specifically chosen for its low-power consumption and sub-1 GHz capabilities, enabling efficient, long-distance communication.

Fig. 2(b) showcases a photograph of the implemented communication module, highlighting the physical realization of our hardware setup.

The block diagram of the Access Point (AP) is presented in Fig. 2(c). The AP is constructed using a Raspberry Pi along with a communication module running AP-specific software. The Raspberry Pi acts as a data relay between the Network Management Server (NMS) and the AP's communication module via an Ethernet connection. Within the AP, the Raspberry Pi and the communication module is connected through a UART (Universal Asynchronous Receiver–Transmitter) interface with a baud rate of 921,600 bits per second.

Fig. 2(d) illustrates the software structure of the communication module, encompassing the DSME (Deterministic and Synchronous Multi-channel Extension) Medium Access Control (MAC) layer and the proposed Over-the-Air (OTA) firmware update mechanism. This structure also includes essential functional blocks such as Hardware Abstraction Layer (HAL), Transmission (TX) and Reception (Rx) Engine, and other supporting components essential for IoT networking.

### 3.3. LoRa PHY

In this paper, the LoRa (Long Range) transmission technique is chosen as the Physical (PHY) layer to be used in conjunction with the DSME MAC. LoRa is a proprietary modulation system developed by Semtech Semiconductor that uses Chirp Spread Spectrum (CSS) signals to reduce interference. The CSS enables reliable communication even in very noisy environments. It also consumes less power because it does not require complex signal processing. Using parameters such as spreading factor (SF), coding rate (CR), and bandwidth (BW), LoRa radio can change the transmission distance, speed, and reliability.

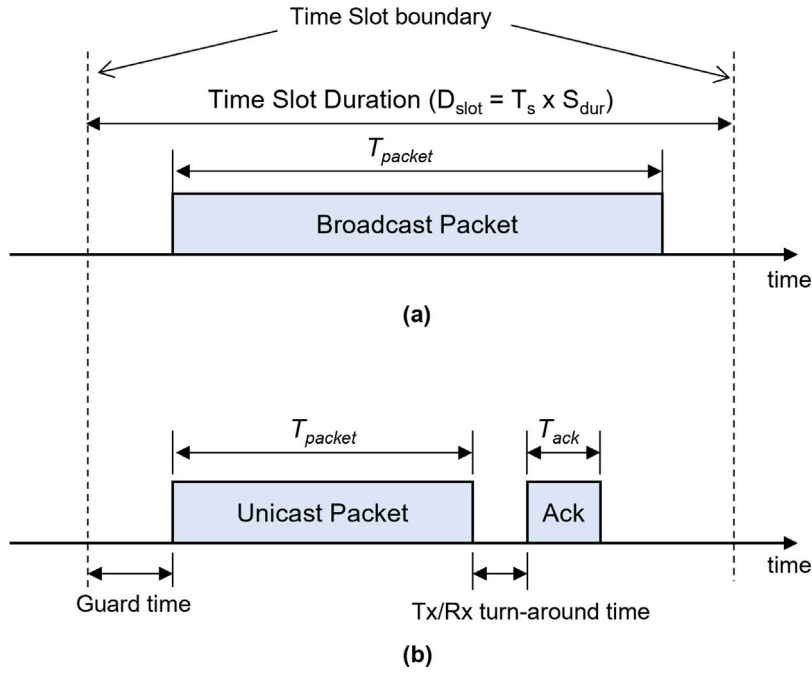


Fig. 3. Structure of Time Slots (a) when Ack is not required (Beacon or GTS for OTA Download) and (b) when Ack is required (CAP or GTS).

The network proposed in this research is a multi-hop network that uses mid-range communication as an intermediate concept between LR-WPAN and LPWAN. This approach not only allows for the extension of network coverage but also minimizes shadow areas while maintaining a sufficiently high transmission rate.

Similar approaches can be found in other articles. For example, in [52] the authors propose a technique for using DSME together with LoRa to overcome the application limitations of LoRaWAN caused by various limiting factors.

### 3.4. Time slot duration and Beacon Interval

The structure of the three different time slots (Beacon, CAP, and GTS) is illustrated in Fig. 3. During Beacon or OTA download in GTS slots, as shown in Fig. 3(a), broadcast transmissions are used, eliminating the need for acknowledgment (Ack) from receiving nodes. In contrast, transmissions in CAP or GTS slots utilize unicast transmissions, prompting the receiver to send an Ack packet back to the sender. Such case is illustrated in Fig. 3(b).

The time slot structure, as depicted in the figure, with dashed lines indicating slot boundaries, is composed of several components: The Guard Time is the duration allocated at the beginning of each time slot to ensure proper separation between consecutive transmissions, minimizing interference and allowing for signal settling. The  $T_{packet}$  (Time on Air) refers to the duration required for a LoRa frame to be transmitted over the air. The Tx/Rx Turn-around Time is the time required for a device to switch between transmit (Tx) and receive (Rx) modes. After transmitting a packet, the device needs to switch to receive mode to listen for acknowledgments or incoming transmissions. The  $T_{ack}$  (Acknowledgment Time) is applicable to unicast transmissions in CAP or GTS slots, which represents the time taken for a receiving node to send an acknowledgment (Ack) packet back to the sender, confirming successful receipt of the transmitted data.

In Eqs. (1)–(4), the duration values are expressed in symbol units. To convert these symbol durations into actual time durations (Time Slot Duration,  $D_{slot}$ ), we utilize the Symbol Duration ( $S_{dur}$ ). Throughout this study, we adopt Symbol Duration ( $S_{dur}$ ) values of 14 and 22 microseconds per symbol ( $\mu s/symbol$ ), corresponding to Time Slot Durations ( $D_{slot}$ ) of 26.8 and 42.2 milliseconds (ms), respectively. These durations result in Beacon Intervals (BIs) of 6.88 and 10.81 seconds (s), respectively. Additionally, we calculate the time on air ( $T_{packet}$ ), which represents the duration a LoRa frame takes to transmit, considering the chosen modulation parameters and payload size.

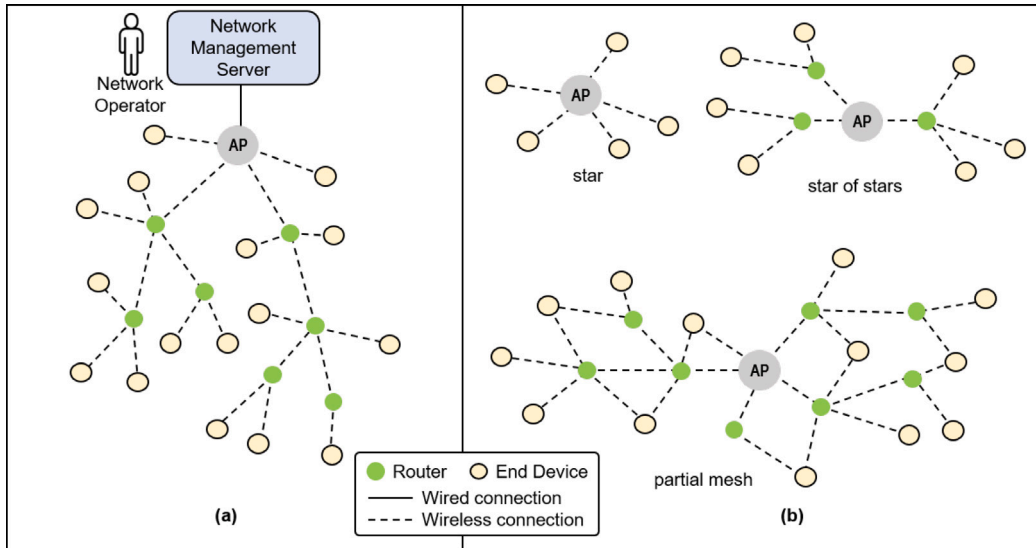
We also employ the configuration of  $BO = MO = 9$  and  $SO = 5$ , resulting in a total of 112 transmission slots (either CAP or GTS) within each Beacon Interval (BI), as detailed in Table 2.

### 3.5. Multi-hop network structure

In order to eliminate the shadow area and increase the wireless coverage, this paper constructs a multi-hop IoT wireless network based on the cluster-tree topology [53], as shown in Fig. 4(a). This network consists of a network management server (NMS) and

**Table 2**  
Symbol & slot duration and beacon interval.

Symbol duration ( $\mu\text{s}/\text{symbol}$ )	Time slot length (symbols)	Slot duration (ms)	Beacon Interval (s)	Number of time slots in BI (BO = MO = 9, SO = 5)
14	1920	26.8	6.88	112
22	1920	42.2	10.81	112



**Fig. 4.** Cluster-tree based multi-hop network topology (a) and some variations like a star, a star of stars, and a partial mesh (b).

various wireless components, including an Access Point (AP) and one or more routers or end devices that support the IEEE 802.15.4 DSME-based multi-hop IoT wireless network. Fig. 4(b) shows some variations such as star, star of stars, and a partial mesh network. In the figure, the network operator is the one who is responsible for preparing the firmware update and delivering it to the NMS.

In the figure, the AP is a node that receives and handles association requests from terminal nodes, and also broadcasts periodic Beacon packets to devices that are requesting to connect to the network. The beacon contains slot synchronization information, time slot allocation state, and network superframe structure.

The router acts as an intermediary device that relays information between end devices and an Access Point (AP) to facilitate the collection of sensor data. One of its key responsibilities is to relay Beacon packets sent by the AP to ensure they reach all end devices (EDs). Routers can be configured as Full Function Devices (FFDs) according to the IEEE 802.15.4 standard.

The end device is a node that collects sensor data. The data is passed to the NMS via one or more routers and the AP.

## 4. OTA implementation

### 4.1. Overview of DSME-FOTA

In this research, we propose a multi-hop based wireless firmware update to simplify the maintenance of a large number of IoT devices. In our implementation, the NMS initiates the deployment and application of new firmware updates wirelessly to each node, including AP and router, via a multi-hop IoT wireless network.

The following are the characteristics of the framework:

- Firstly, it provides a way to ensure that all nodes selected by the NMS receive the updated firmware. The wireless firmware update may include a critical update such as a change in wireless channel or transmission bandwidth. A firmware mismatch will result in a communication or service failure if the update is not applied to any of the routers or end devices. In this case, it may result in an orphaned device.
- Second, the superframe structure is adapted as necessary to include time slots for uplinks and/or downlinks for efficient wireless updates. It is preferable to divide the firmware into many blocks and transmit them because the length of the firmware to be updated typically exceeds the data length that can be transmitted at once in the IoT network. When the number of firmware blocks to be transmitted is enormous, the strategy proposed in this work reduces the time required for the full firmware transmission by reallocating the GTS slots within the superframe for uplink or downlink transmission as required.

## 4.2. FOTA master and slave functions

To implement the wireless firmware updates, each wireless node in the IoT wireless network has OTA master or OTA slave functionality. There is a parent–child relationship between a node and the child nodes connected to it in the network. The OTA master function is active in the parent node, while the OTA slave function is active in the child node.

The proposed framework uses a repeated cascade arrangement of master–slave operations to distribute new firmware throughout the network. In the case of a router, it first acts as an OTA slave, downloading firmware from its parent node. It then switches to OTA master and broadcasts firmware to its children.

## 4.3. FOTA framework

Fig. 5 shows the whole process of wireless firmware update in the multi-hop based IoT wireless network proposed in this paper. The numbers within the circles in the figure correspond to each step of the FOTA process. A detailed explanation of each is given below.

**Step ①:** The NMS sends firmware metadata and firmware binary to the Access Point (AP) to begin the wireless firmware update. The AP sends acknowledgment (Ack) packets for each piece of data to indicate whether or not the transmission was successful or not. Upon successful reception, the NMS issues an OTA Start message to the AP in order to initiate the OTA process.

**Step ②:** After the AP receives the OTA Start message, it starts to send OTA Sleep Beacon<sup>1</sup> packets periodically to all wireless nodes informing them that it has switched to OTA mode and has begun the firmware download procedure. In a multi-hop configuration, the router forwards the packet to its neighbors. The wireless nodes that receive this packet will switch to OTA mode and wait in the Sleep state. This procedure continues for at least a pre-defined amount of time to ensure that each wireless node can successfully receive the Beacon or relayed Beacon packet.

**Step ③:** This step consists of three states Ready, Download, and Resend. First, the AP takes on the role of an OTA master by starting to send the OTA Ready Beacon packet, which tells its neighbors (routers and end devices) to switch to the Ready state and verifies the transition. After each child node receives this Beacon packet, it sends a OTA Ready Response packet to the parent node to notify that it is ready to download the firmware. The OTA master proceeds to the next step after receiving the OTA Ready Response from each child node. Second, the OTA master broadcasts firmware blocks, and the child nodes that receive them use Cyclic Redundancy Check (CRC) to verify the downloaded blocks. Thirdly, the OTA master enters the Resend state after the transmission is complete. In this state, if any of the received blocks are missing or if an error is found in the CRC check, the child node requests retransmission of the missing block to the parent node. The retransmission procedure of this Resend state is repeated until no more retransmission requests are received. A child node that receives all blocks in the Download state or through the Resend state will continue the CRC verification for the complete download.

**Step ④:** When no more resend requests are received from child nodes, the OTA master begins to send a OTA Complete Beacon packet to indicate that its download procedure is complete. If there are one or more routers among the child nodes, the Beacon packet will contain the NextHopMaster value corresponding to one of the relay node addresses. The OTA master transmitting the OTA Complete Beacon packet with this NextHopMaster value serves the following purpose:

- The router that corresponding to the address is given permission to begin its own firmware download procedure for the neighborhood.
- Any other (if any) routers that do not match will be forced to wait until they receive the OTA Complete Beacon packet corresponding to their address.
- Instruct the end devices to wait until they receive the OTA Complete Beacon.
- This packet also has the additional purpose of informing the parent of the OTA master that the firmware download from a child is now in progress.

**Step ⑤:** This step reflects the firmware download procedure, which is initiated by a newly selected router. Similar to Step ③, the procedure is repeated, but this time a new set of nodes is used for the master/slave connection between the OTA entities. The procedure of sequentially selecting such relay nodes and downloading the firmware through the relay node continues until the firmware of all wireless nodes in the multi-hop IoT wireless network has been updated.

**Step ⑥:** An AP or router acting as an OTA master indicates that there are no remaining routers among its child nodes for OTA downloads by setting the NextHopMaster value in the OTA Complete Beacon packet to 0xFFFF. When the OTA Master changes the NextHopMaster value in the OTA Complete Beacon packet or receives an OTA Complete Beacon packet transmitted by the child relay node, it notifies the NMS by sending the OTA Complete Notify packet as well.

**Step ⑦:** The AP concludes that the firmware download process is complete for all wireless nodes in the network is complete when it finishes the firmware download for all AP's child end devices of the AP and receives the OTA Complete Beacon with the NextHopMaster value 0xFFFF from all child routers of the AP. In this case, the AP will initiate the firmware installation procedure so that all wireless nodes can apply the updated firmware. To instruct the firmware application, the AP broadcasts the OTA Finish Beacon for a certain period of time.

<sup>1</sup> An OTA STATE Beacon packet is a Beacon packet containing information about the OTA operation in STATE state, where the STATE is one of the following states: Sleep, Ready, Download, Resend, Complete, and Finish.

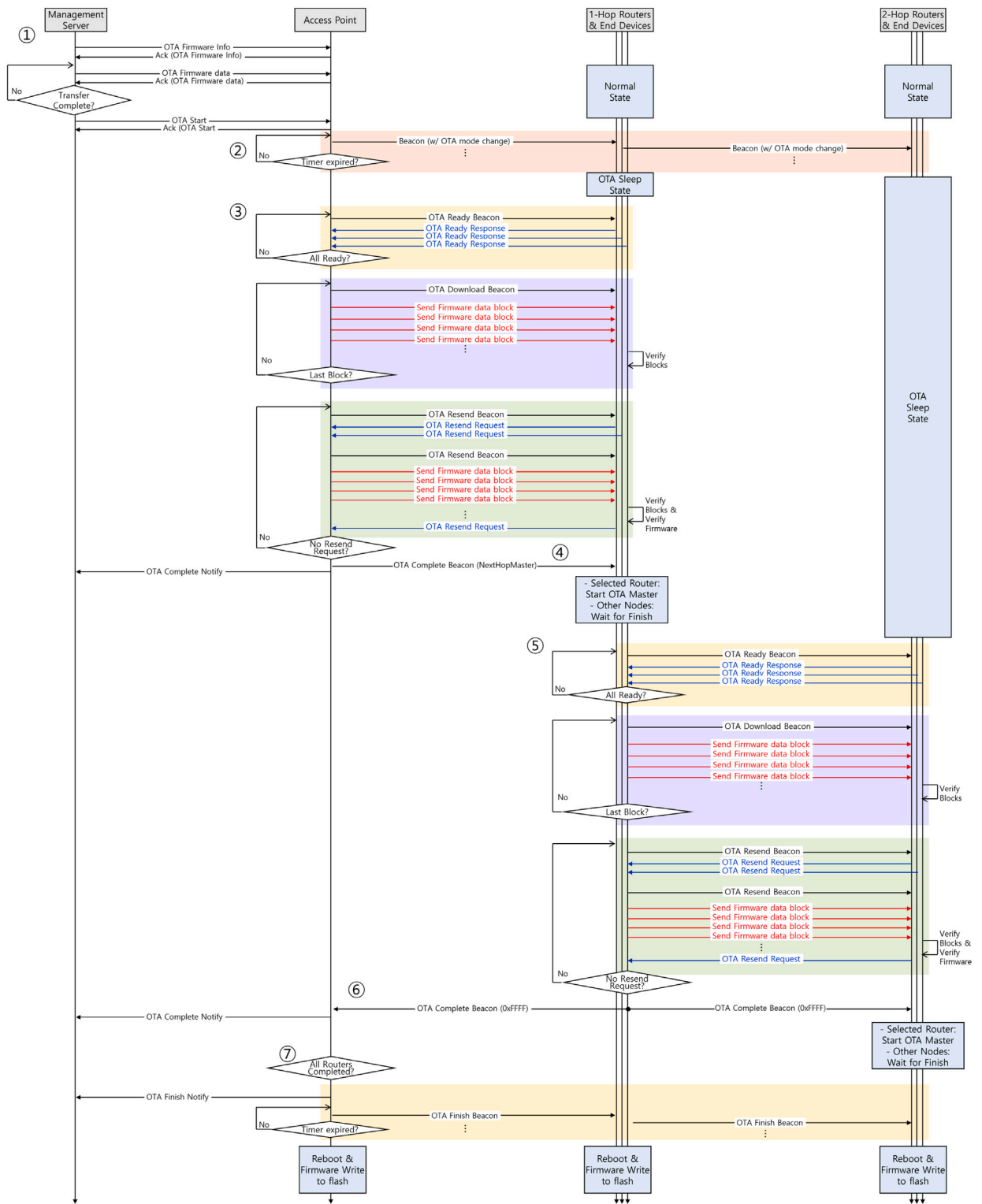


Fig. 5. Overall FOTA process.

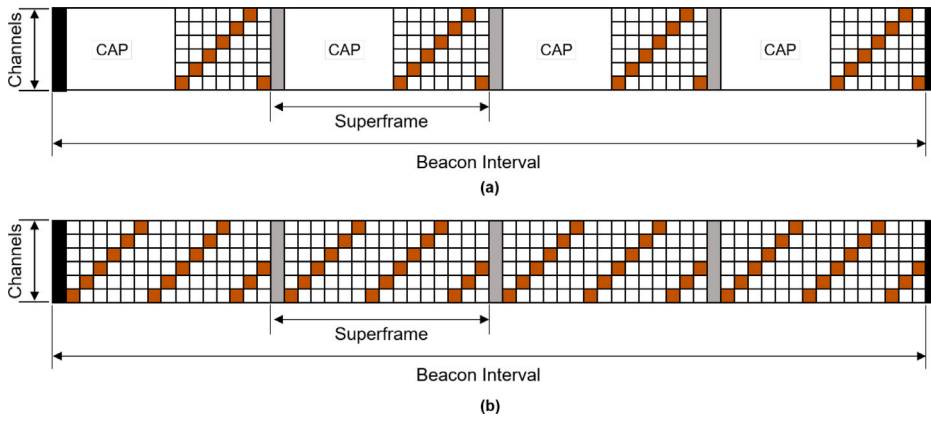


Fig. 6. Superframe structure and CH sequences (a) in normal DSME operation (i.e., sensing), (b) in FOTA downloading.

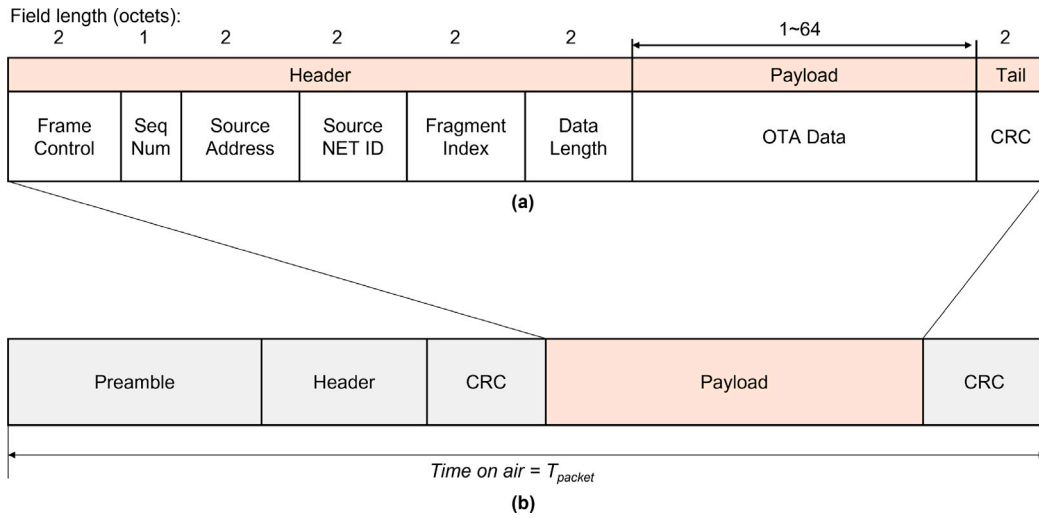


Fig. 7. Data packet structure: (a) FOTA Download packet format encapsulated within (b) LoRa frame structure.

#### 4.4. Reallocation of superframe structure

According to the IEEE 802.15.4 standard, one radio node can be assigned one or more GTS slots inside a single superframe. To reduce the time required for the firmware download, all GTS slots in the superframe are used for the downlink and allocated to the OTA master so it can transmit firmware blocks. The slot allocation and CH sequences for normal DSME operation and FOTA are shown in Fig. 6 for comparison.

#### 4.5. Packet format

In the context of Over-The-Air (OTA) firmware updates using LoRa networks, the firmware binary is managed in 256-byte segments, aligning with the flash memory write size. However, due to the transmission constraint of Guaranteed Time Slots (GTS) where each slot can accommodate a maximum of 64 bytes, each firmware segment is divided into four 64-byte fragments for efficient transmission. This division optimizes the use of GTS slots, crucial for timely and reliable OTA updates.

Each OTA download packet consists of a 11-byte header for metadata, followed by a firmware fragment of up to 64 bytes and a 2-byte Cyclic Redundancy Check (CRC) value to ensure data integrity. The packet format, illustrated in Fig. 7(a), is designed to fit seamlessly within the Slot Duration setup of GTS, allowing for efficient utilization of available transmission slots.

Furthermore, to integrate OTA download packets into the LoRa communication framework, each packet is encapsulated within the LoRa frame format as depicted in Fig. 7(b). The LoRa frame structure comprises several key components: First, the preamble consists of a sequence of symbols used for synchronization between the transmitter and receiver. Following the preamble, the LoRa header contains metadata such as payload size, coding rate (CR), and a CRC for integrity verification of the header. The payload

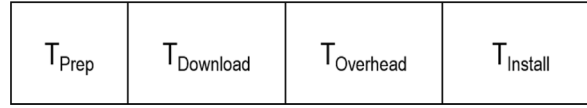


Fig. 8. Breakdown of the temporal aspects of the FOTA process.

section contains the actual data being transmitted, including firmware fragments in the case of OTA download packets. Finally, at the end of the frame, a CRC value is appended to verify the integrity of the entire frame.

## 5. Performance analysis

In this section, we present a comprehensive numerical analysis of the overall performance, with a special focus on the FOTA update time. This provides a quantitative assessment of the effectiveness of the framework. These findings are invaluable for predicting expected FOTA update times when implementing DSME-FOTA in various network configurations.

### 5.1. Breakdown of the FOTA process

For the design of the DSME-FOTA, an essential metric to emphasize is the time required for the entire process, which can be broken down into four distinct components, as shown in Fig. 8.

The definitions for each of the part are as follows:

- The  $T_{Prep}$  period in this figure represents the time it takes for the NMS to send a firmware binary and FOTA information to the Access Point (AP), including the time for the AP to write the firmware binary to the flash memory of the communication module.
- The  $T_{Download}$  portion represents the amount of time necessary for the OTA master to wirelessly broadcast the firmware to its neighbors (i.e., the OTA Slaves). In the download phase, the OTA Slaves are also required to write firmware blocks to flash memory. However, it is important to note that in sensor nodes it is typically not possible to receive RF and write to flash memory at the same time. This requires a separate time allocation for the writing process. Consequently, the duration of this phase varies significantly depending on the size of the firmware (see 5.2.4).
- The  $T_{Overhead}$  section is the cumulative time required by the OTA master to ensure that no end device is left out of the update and that none remains in an unsynchronized state. It also includes the time taken to request retransmission due to a transmission error and the time required to retransmit the firmware block during the Resend state.
- The  $T_{Install}$  section specifies the time required for the OTA slave to write the firmware information to the flash memory and attempt a reboot. During the reboot, the bootloader software checks for updated firmware information and copies the new firmware binary to the location where the original firmware resides.

The first and last sections shown in Fig. 8 are strongly tied to a particular implementation, although the second and third indicate how efficient the framework is. For this reason, this article carefully examines how long the Download and the Overhead parts take.

### 5.2. Download and overhead analysis

To analyze the time durations  $T_{Download}$  and  $T_{Overhead}$ , we use a network as shown in Fig. 9. This figure represents a scenario where one AP and eight routers have formed a cluster-tree based multi-hop network.

In this figure, we denote the neighbors of the AP (red circle) and R1 (blue circle). It is important to note that in this paper, the neighbor relationship is not solely dependent on radio range, but are also determined by network connectivity results. For example, in the figure, Node A is in close proximity to the AP, but it may be advantageous for it to connect to a router due to signal strength considerations.

The download pipelining approach, as described in [54], is not used in our implementation. To apply pipelining, it is necessary to ensure that when two source nodes start transmitting at the same time, they do not interfere with any other nodes that are located at an intermediate point between them. However, solving the hidden node problem is expected to increase the complexity significantly. Therefore, each neighborhood on the network proceeds with firmware downloads one at a time without the use of pipelining.

Let us denote the probability of a single packet (a firmware fragment block) transmission failure<sup>2</sup> as  $p$ . In general, this probability is assumed to be the same for all nodes. However, in wide-area networks or networks operating in harsh environments, this probability can be expected to increase with the distance of the nodes from the transmission source. The configuration of multi-hop networks aims to address this issue and keep the  $p$  value within a certain range.

Since it is necessary to deliver firmware blocks to all nodes in the network without omission, even if the probability  $p$  is greater than zero, it is necessary to retransmit the firmware blocks, resulting in a corresponding increase in  $T_{Download}$  time.

<sup>2</sup> For firmware blocks transmitted via broadcast, a failed transmission is defined as a scenario in which at least one of the nodes in the network fails to receive the block. In contrast, a successful transmission, or a "success", is defined when all nodes successfully receive the block.

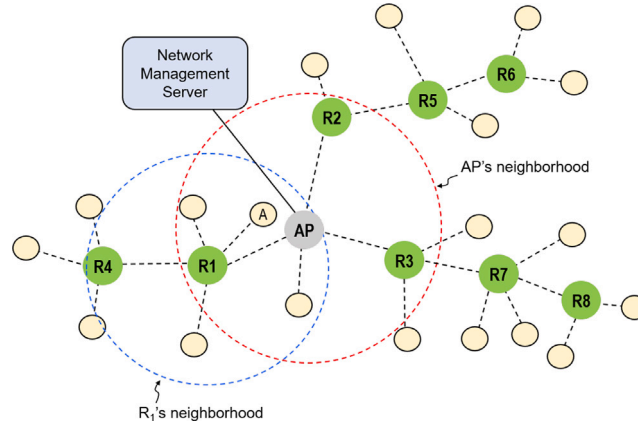


Fig. 9. A multi-hop network scenario.

Table 3

Notation table.

Notation	Description
$\mathbb{R}$	The set of neighborhoods. The size is denoted as $R =  \mathbb{R} $
$r$	A neighborhood, $r \in \mathbb{R}$
$N_r$	The total number of nodes in a neighborhood $r$
$\alpha$	The number of nodes that successfully received the packet during a failed transmission attempt
$p$	The probability of an individual node fail to receive (general)
$p_r$	The probability of an individual node fail to receive in a neighborhood $r$
$E(N_r, p_r)$	The expected number of transmissions needed to successfully deliver the packet to all $N_r$ with the given $p_r$
$S$	Firmware binary size, or more precisely, the number of fragment blocks into which the firmware is divided
$D_{\text{slot}}$	Duration of a slot ( $\mu\text{s}/\text{symbol}$ )
$k$	Transmission efficiency
$F$	The number of fragments in a segment
$W$	The number of waiting slots
$CT_x$	The number of times a single block is transmitted (repeatedly)

### 5.2.1. Analytical model

To analyze the number of transmissions that occur within a single neighborhood, we use the notation in Table 3.

The expected number of transmissions of a firmware block in neighborhood  $r$  to ensure that all nodes in the neighborhood can receive the block is given as:

$$E(N_r, p_r) = 1 + p_r \cdot E(N_r - \alpha, p_r), \quad (5)$$

where the term ‘1’ represents current transmission attempt,  $p_r$  represents the probability of a *failed* transmission in neighborhood  $r$ , and the term  $E(N_r - \alpha, p_r)$  represents the expected number of transmissions needed for the subset of nodes that did not receive the packet ( $N_r - \alpha$  nodes) in the next transmission. If the firmware is divided into  $S$  fragments for transmission, then the expected number of downloads (transmissions) required in neighborhood  $r$  is as follows:

$$E[T_{x_{\text{Download of } r}}] = S \cdot E(N_r, p_r) \quad (6)$$

Therefore, the total expected number of FOTA downloads required for all  $R$  neighborhoods existing in the network is:

$$E[T_{x_{\text{Total}}}] = \sum_{r=1}^R E[T_{x_{\text{Download of } r}}] \quad (7)$$

$$= \sum_{r=1}^R S \cdot E(N_r, p_r) \quad (8)$$

GTS slots are used for the download process, but as previously mentioned, the total FOTA download time can be further increased by the probability  $p_r$  of errors occurring in the neighborhood of  $r$  and flash write times (see Section 5.2.4).

### 5.2.2. Number of transmissions in a neighborhood

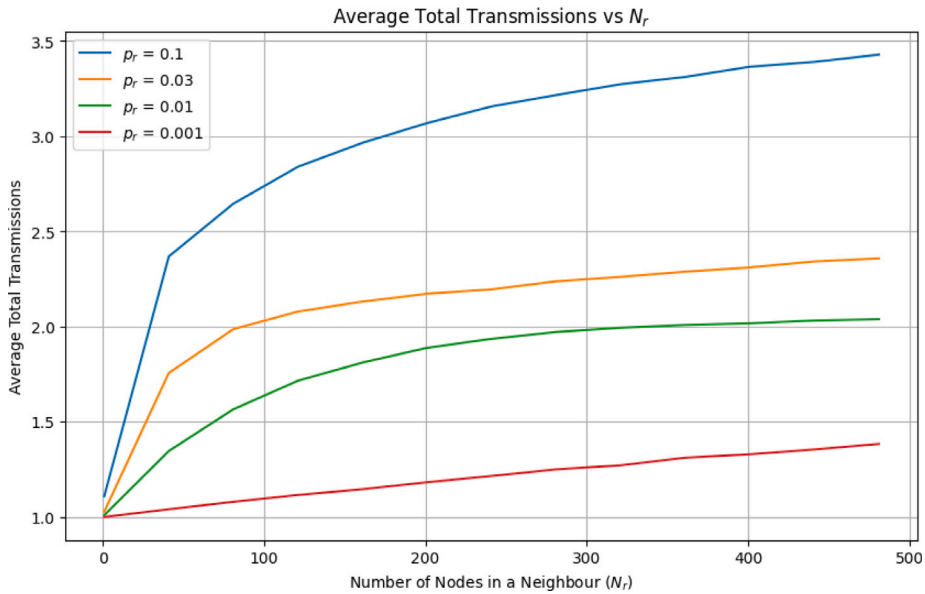
Due to the recursive nature of Eq. (5), a simulation-based approach was used to further investigate the number of transmissions in a neighborhood. The *pseudo-code* for the simulation algorithm used in this analysis is given in Algorithm 1.

**Algorithm 1** The number of transmissions for given  $N_r$  and  $P_r$  in a neighborhood  $r$  using repeated trials.

```

1: procedure SIMULATENUMBEROFBROADCAST( $N_r, p_r, numTrials$ ):
2:   totalTransmissions = 0
3:
4:   for  $j$  from 1 to  $numTrials$  do
5:     transmissions = 0
6:     success = False
7:     nodesNotReceived =  $N_r$ 
8:
9:     while not success do
10:      transmissions += 1
11:      successfulNodes = 0
12:
13:      for  $i$  from 1 to nodesNotReceived do
14:        if  $randomNumber() > p_r$  then
15:          successfulNodes += 1
16:        end if
17:      end for
18:
19:      nodesNotReceived -= successfulNodes
20:      if nodesNotReceived == 0 then
21:        success = True
22:      end if
23:    end while
24:
25:    totalTransmissions += transmissions
26:  end for
27:
28:  averageTransmissions = totalTransmissions/ $numTrials$ 
29:  Return averageTransmissions
30: end procedure

```



**Fig. 10.** Average number of transmissions to deliver a single packet to  $N_r$  nodes in a neighborhood.

**Fig. 10** illustrates the average number of transmissions required to send a single packet to  $N_r$  nodes within a single neighborhood, where the probability of an individual node within a neighborhood  $r$  failing to receive a transmission is given as  $p_r$  (assuming that the receive failure probability is equal and independent for each node<sup>3</sup>).

The graph shows that even if the network itself is not completely stable ( $p_r = 0.1$ ), the average number of transmissions does not increase significantly as the number of nodes grows. For instance, with an error rate of 10% ( $p_r = 0.1$ ), with  $N_r = 50$  nodes,

<sup>3</sup> In the multi-hop network setup, the distance from an end device to the nearest router (or the Access Point) it is connected to is typically less than the radio range. This setup helps ensure successful transmission and reduces the heterogeneity of the IIoT environment in terms of signal quality.

the average number of transmissions is about 2.45 times. With  $N_r = 100$  nodes, the average number of transmissions increases to about 2.75 times. This result can be used to determine the appropriate number of retransmissions based on the values of  $N_r$  and  $p_r$ .

### 5.2.3. Download time ( $T_{Download}$ )

As shown in Fig. 10, for example, if  $N_r$  (the number of nodes in a single neighborhood) is less than or equal to 100 and  $p_r$  (the probability of transmission failure for each node) is less than 0.03, then only two transmissions of each block are sufficient to achieve a high probability of successful reception at all nodes in the network. This suggests the possibility of simplifying the term  $E(N_r, p_r)$  in the Eqs. (6) and (8) by replacing it with a fixed value,  $CTx$  (a constant), as follows:

$$Tx_{\text{download of } r} = S \cdot CTx \quad (9)$$

$$Tx_{\text{total}} = \sum_{r=1}^R (S \cdot CTx), \quad (10)$$

where  $CTx$  represents the number of times a single block is transmitted (repeatedly). Since a fragment of a segment is transmitted in each time slot, each neighborhood requires a time corresponding to  $S \cdot CTx$  time slots.

When multiple routers are present in a neighborhood (as shown in Fig. 9, where the AP's neighborhood includes  $R1$ ,  $R2$ , and  $R3$ ), these routers take turns acting as OTA Masters and proceed with OTA downloads.

Given that Eq. (10) calculates the total number of transmissions required for all neighborhoods, we can use it to determine the  $T_{Download}$  value. It is important to consider the transmission data rate, which is directly influenced by the LoRa PHY model. In our implementation, we assume that each slot, following the time slot structure in Fig. 3, transmits a single fragment block. Therefore, the transmission time for each slot is ultimately determined by the length of a single slot, denoted as  $D_{\text{Slot}}$ . This slot duration reflects the characteristics of the LoRa PHY model, including factors such as spreading factor, bandwidth, and coding rate, which collectively affect the achievable data rate (and accordingly the  $T_{\text{packet}}$  or the  $T_{\text{ack}}$ ).<sup>4</sup> With this in mind, the total download (transmission) time can be expressed as follows:

$$T_{\text{Download}} = [R \cdot S \cdot CTx] \cdot D_{\text{Slot}} \quad (11)$$

### 5.2.4. Consideration of flash memory access time

During the implementation of IoT networks using embedded systems and the design of the FOTA framework, we found that a challenging condition occurs during the process of storing received 256 byte segments of the firmware binary to flash memory. This operation can potentially block the transmit/receive (tx/rx) process, resulting in a condition where incoming subsequent OTA blocks cannot be received while flash memory writes are in progress [55].

To address this issue, the transmitter (i.e., the OTA master) should schedule its data transmissions on the GTS slots accordingly. This scheduling may involve introducing intentional delays between successive transmissions to allow sufficient time for flash memory write operations at the OTA slave. The problem actually occurs when two conditions are present at the same time:

- The flash memory write time is significantly longer than the OTA Download slot length; and
- The embedded IoT device has a limited memory size, that requires immediate writes to flash memory each time a firmware block is downloaded. (In our case, this operation is performed each time the fourth fragment is received.)

In general, such scheduling in the transmitter is outside the scope of the FOTA framework, but it must be considered because it affects the overall FOTA performance.

To begin with the study, it is necessary to determine how long the flash memory writing process takes. Using our implemented hardware with an STM32L162 microcontroller (refer to Section 3.2), the flash memory write time for several selected binary sizes (8, 16, 32, 64, 128, 256, and 512 bytes) is shown in Fig. 11. It is measured on the OTA slave during the OTA Download phase, with arbitrary binary data written to the flash.

On the right-hand side of Fig. 11, the durations of some multiples of the time slots are shown for reference when the Symbol Duration ( $S_{dur}$ ) is set to 22  $\mu\text{s}$ . As can be seen from the figure, the average delay times show a linear increase as the binary size increases. Since the STM32L162 has a page size of 256 bytes, it becomes apparent that 6 or more time slots are required to write 256 bytes to the flash memory. This minimum slot requirement imposes a constraint on OTA download performance because the superframe has only 15 GTS slots.

To account for the flash memory write delay during the OTA Download phase at the transmitter, two representative slot pattern examples are shown in Fig. 12. In this figure, 'F' represents slots for data fragments.<sup>5</sup> 'W' denotes wait slots — slots intentionally left for flash memory write operations. Fig. 12(a) illustrates a scenario where  $\text{Frag} = 4$ ,  $CTx = 1$ , and  $W = 11$ . In Fig. 12(b),  $CTx$  is configured as 2, leaving the remaining 7 slots for flash memory write operations.

The slot pattern shown in Fig. 12 was used for the DSME-OTA implementation in this paper. Particularly, even when choosing  $CTx = 2$ , there is an observed increase in the probability of successful transmission compared to  $CTx = 1$ , especially when the  $p_r$  value is lower than 0.03, without an accompanying increase in the  $T_{\text{FOTA}}$  value.

<sup>4</sup> The use of slot duration simplifies the analysis by hiding the details of the LoRa PHY model, providing a convenient framework for modeling transmission efficiency.

<sup>5</sup> Each fragment contains 64 bytes of the firmware, and four fragments collectively form a 256-byte page size.

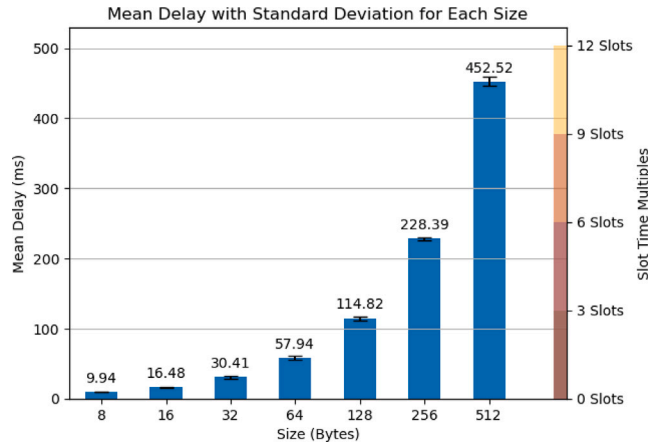


Fig. 11. Average flash memory write time for different data sizes.

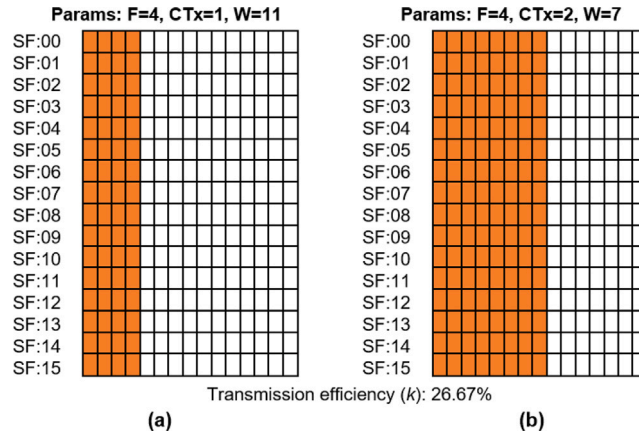


Fig. 12. Slot patterns during FOTA Download procedure. (a)  $F = 4$ ,  $CT_x = 1$ ,  $W = 11$ , and (b)  $F = 4$ ,  $CT_x = 2$ ,  $W = 7$ .

Let  $k$  be the transmission efficiency, meaning that only  $k\%$  of the slots are actively used for transmission (when  $CT_x = 1$ ). Then the expected download time  $T_{Download}$  in Eq. (11) increases accordingly, since it is proportional to the expected number of transmission slots. So the equation is rewritten as follows:

$$T_{Download\ of\ r} = (100/k) \cdot [S \cdot CT_x] \tag{12}$$

$$T_{Download} = T_{Download\ of\ r} \cdot R = (100/k) \cdot [R \cdot S \cdot CT_x] \tag{13}$$

### 5.2.5. Overhead time ( $T_{Overhead}$ )

In Eq. (11), we noted that the download time ( $T_{Download}$ ) is determined by factors such as the number of neighborhoods, fragment blocks, and retransmission attempts. In contrast, the overhead time ( $T_{Overhead}$ ) includes the time required to ensure that all nodes in the network are in the same state. In a DSME network, all slots repeat in cycles of Beacon Intervals (BIs). Therefore, setting the overhead time as a multiple of the BI period is a natural choice. We use the following equations:

$$T_{Resend} = (3 \cdot BI) \tag{14}$$

$$T_{Ready} = T_{Complete} = T_{Finish} = (4 \cdot BI) \tag{15}$$

The time at which the OTA master initiates this state transition may not necessarily synchronize with the Beacon Interval (BI). However, the OTA slave begins the transition the moment it receives the Beacon packet from the OTA master, which includes information about this transition.

Both  $T_{Ready}$  and  $T_{Resend}$  are repeated for each neighborhood. When the download is complete for all neighborhoods,  $T_{Complete}$  and  $T_{Finish}$  will follow.

Therefore, if there are  $R = |\mathbb{R}|$  neighborhoods in the network, the approximate  $T_{Overhead}$  time can be calculated as follows:

$$T_{Overhead} = R \cdot (T_{Ready} + T_{Resend}) + T_{Complete} + T_{Finish} \tag{16}$$

### 5.3. Overall FOTA update time

Calculating the total OTA time in Fig. 8 using Eq. (13) and (16) yields the following result (except for the  $T_{Install}$ ). It is important to note that the  $T_{Prep}$  exists only in the neighborhood of AP and is equivalent to the  $T_{Sleep}$  in terms of timing. This will be explained in more detail in the following discussion (particularly in Fig. 14).  $T_{Download}$  represents the download time from all neighborhoods.  $T_{Overhead}$  is the overhead time when there are  $|\mathbb{R}|$  neighborhoods. The total time is as follows:

$$T_{FOTA} = T_{Prep} + T_{Download} + T_{Overhead} \quad (17)$$

The procedure for calculating the total  $T_{FOTA}$  using  $T_{Overhead}$  and  $T_{Download}$  is shown in the Algorithm 2. Since the value of  $T_{Prep}$  varies with the binary size of the firmware, we use a measurement approach across different firmware sizes to construct a linear regression graph (the *slope* and the *intercept*) obtained from the test setup in Section 6.

---

#### Algorithm 2 Pseudo-code for calculating $T_{FOTA}$ values.

---

```

1: procedure CALCULATE_TDOWNLOAD(fileSize)
2:   S := number of fragments from fileSize
3:   TDownload := (100/k) * (S * CTx) * Dslot
4:   Return TDownload
5: end procedure
6:
7: procedure CALCULATE_TPREP(fileSize)
8:   slope := 0.001428
9:   intercept := 9.5
10:  Return (slope * fileSize + intercept)
11: end procedure
12:
13: procedure CALCULATE_TFOTA(fileSize, R)
14:  TDownload := calculateTDownload(fileSize)
15:  TResend := 3 * BI
16:  TReady := TComplete := TFinish := 4 * BI
17:  TOverhead := R * (TReady + TResend) + TComplete + TFinish
18:  T_FOTA := calculateTPrep(fileSize) + TDownload * R + TOverhead
19:  Return T_FOTA
20: end procedure

```

---

It should be noted that in Fig. 12, the value of the transmission efficiency ( $k$ ) is measured to be 26.67%, which means that the system is not very efficient. In the system implementation used in this paper (explained in Section 6), two considerations have been taken into account within the DSME MAC superframe structure: (1) the number of wait slots ( $W$ ) must be set to 6 or higher to allow sufficient wait time to write the firmware binary to the flash memory, and (2)  $CTx$  is set to 2 to increase the probability of a successful transmission, particularly when  $p_r$  is greater than zero. For example, if a system requires a very short flash write time (when compared with the Slot Duration ( $D_{Slot}$ )), it is possible to minimize the value of  $W$ , thereby significantly increasing  $k$ . However, it is important to remember that the Beacon slot and the multi-superframe structure must always be included in these calculations.

### 5.4. Duty cycle consideration

In general, duty cycle limitations in the sub-1 GHz ISM band vary significantly across regions such as Europe and the United States. For instance, in Europe, the ETSI standards segment the ISM frequency band into specific sub-bands (e.g., K, L, M, N, P, Q) with varying duty cycle restrictions ranging from 0.1% to 10%. On the other hand, the US ISM band imposes a strict dwell time limitation of 400 milliseconds. However, for frequency hopping systems operating in the 902–928 MHz band under FCC regulations (Code of Federal Regulations (CFR) 47 part 15), specific requirements apply based on the bandwidth of the hopping channel. If the 20 dB bandwidth of the hopping channel is less than 250 kHz, the average time of occupancy on any frequency shall not exceed 0.4 seconds within a 20-second period. Conversely, if the 20 dB bandwidth is 250 kHz or greater, the average time of occupancy on any frequency shall not exceed 0.4 seconds within a 10-second period.

In our case, where the bandwidth (BW) of the hopping channel is 500 kHz, our system is required to ensure that the average time of occupancy on any frequency does not exceed 0.4 seconds within a 10 second period to comply with FCC regulations.

To meet the FCC regulations, for the BI = 6.88 case, the allowable transmission time per channel within the BI is calculated to be approximately 0.275 seconds ( $= \frac{6.88 \text{ seconds}}{10 \text{ seconds}} \times 0.4 \text{ seconds}$ ). When utilizing the channel hopping sequence depicted in Fig. 6 (which uses six different channels) for the transmission slot patterns illustrated in Fig. 12, there will be only 1 or 2 transmissions in each channel in each superframe (according to the  $CTx$  value). For simplicity, assuming a 100 byte packet for each FOTA download packet in a LoRa setup with SF = 5, CR = 4/5, and BW = 500 kHz, the transmission time ( $T_{packet}$ ) is typically less than 15 milliseconds.

Therefore, based on these assumptions, the total transmission time in each channel is calculated to be 0.21 seconds ( $= 15 \text{ ms} \times 14 \text{ superframes}$ ) for the 1 transmission case and 0.42 seconds ( $= 2 \times 15 \text{ ms} \times 14 \text{ superframes}$ ) for the 2 transmission case. Therefore, while the 1 transmission case remains compliant with FCC regulations, the 2 transmission case does not meet the occupancy rule within the specified time period. Therefore, we can conclude in BI = 6.88 case, to accommodate  $CTx = 2$ , it requires at least eight different channels during the FOTA download operations.

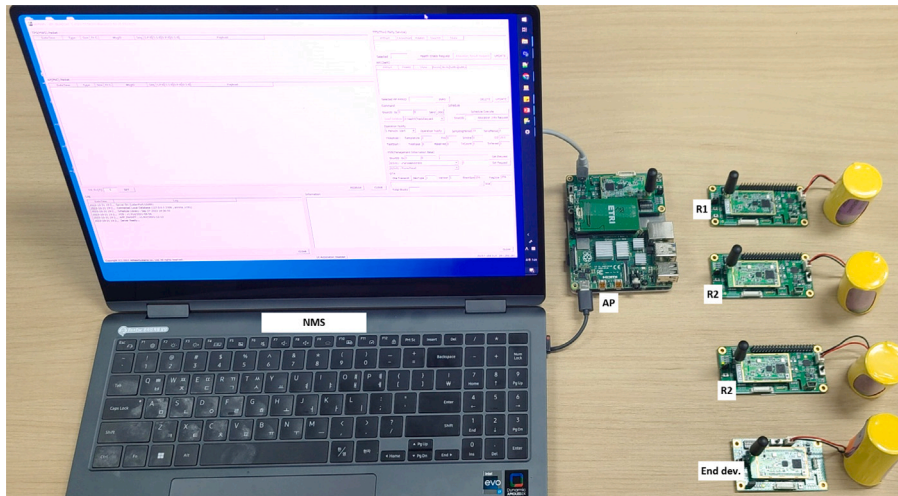


Fig. 13. Testbed setup used to obtain experimental data in the laboratory environment with 4-hop configuration using an AP, three routers, and an end device. The devices were placed in close proximity to each other during the measurements.

## 6. Experiment setup and results

Our experiment serves three purposes. First, we provide timing measurements over the proposed FOTA framework, analyzing the step-by-step timings within a simple two-hop network configuration using the DSME MAC protocol on our testbed. Second, we demonstrate the influence of the firmware binary size and the number of hops (or neighborhoods, more precisely) on the overall FOTA time. Finally, we use these results together with the analytical model presented in the previous section to evaluate the validity of the model.

### 6.1. Testbed

The testbed setup is shown in Fig. 13. Throughout the test, the SX1262 operates at SF = 5, CR = 4/5, and BW = 500 kHz. This results in a data rate of 62 Kbps. The DSME network is set to have SO = 5, MO = 9, BO = 9, and the Symbol Duration ( $S_{dur}$ ) is set to 14  $\mu$ s/symbol when the Beacon Interval (BI) is set to 6.88 seconds, and set to 22  $\mu$ s/symbol when operating at a BI of 10.81 seconds, respectively. The OTA download packet length is set to a maximum of 64 bytes for the payload and 11 bytes for the header. See 6.4 for further discussion on these parameters.

The NMS is a proprietary software that runs on a Windows 10 laptop and includes functionality to monitor the operational status of routers and end devices, and to initiate FOTA by uploading firmware file to the Access Point (AP).

### 6.2. Measurements of elapsed times at each FOTA steps

In the first experiment, elapsed time measurements for FOTA (Firmware Over-The-Air) were performed to verify and evaluate the total and step-wise time intervals. Network configuration parameters were set as follows: BO = MO = 9, and SO = 5. These measurements were performed under two scenarios: one with a Beacon Interval (BI) of 10.81 seconds, and the other with a BI of 6.88 seconds. This change was achieved by adjusting the Symbol Duration to 22 and 14 microseconds per symbol, respectively.

A two-hop network is established using an AP, a router, and an end device (ED). This simple network configuration allows us to accurately observe the FOTA procedures of each node while minimizing the impact that can occur when a large number of nodes are present in the network. The firmware used in this test was 113 KB in size. Time measurements were performed by comparing timestamps extracted from event logs generated by OTA Masters or OTA Slaves in each neighborhood. The measurement results for the scenario with a BI (Beacon Interval) of 6.88 are shown in Fig. 14.

In this figure, the event times recorded at each step are indicated by green vertical lines. For additional clarity and to indicate the relative operational positions at each node, the FOTA procedures are marked with colors. The measured timestamps are shown as relative time values, with the start of the AP (transition to OTA mode) serving as the reference time at 00:00.

According to Fig. 14, when the FOTA is initiated, all nodes in the network, including the AP, go into the Sleep state (it takes almost 3 minutes for AP's neighbors). This time is mostly determined by the time it takes the NMS to transfer the firmware binary to the AP. Note that the speed of the transfer is not very high, even though it is using a wired connection. This is because the AP itself must also write the received firmware binary blocks to the flash memory of the communication module. Moreover, the AP must perform this process while strictly adhering to the Beacon transmission slot, ensuring that it does not overlap with the Beacon transmission time.

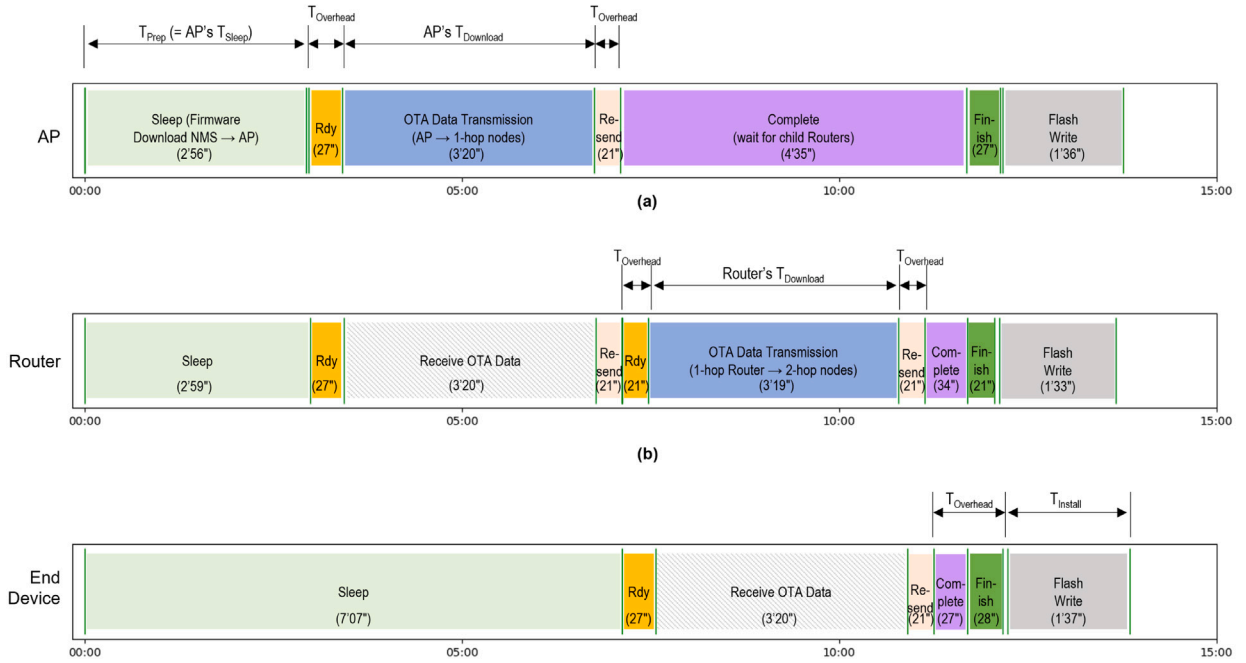


Fig. 14. Elapsed time measurements of the FOTA stages: (a) AP, (b) 1-hop router, and (c) 2-hop end device.

Table 4

Measured times for FOTA for two different Beacon Intervals: BI = 6.88 seconds for Experiment 1 and BI = 10.81 seconds for Experiment 2.

Experiment 1 (BI = 6.88)										
	Prep	Ready	Download	Resend	Complete	-	-	-	Finish	Install
AP	02:56	00:27	03:20	00:21	04:35	-	-	-	00:27	01:36
Router	02:59	00:27	03:20	00:21	00:21	03:19	00:21	00:34	00:21	01:33
ED	07:07	-	-	-	00:27	03:20	00:21	00:27	00:28	01:37
Experiment 2 (BI = 10.81)										
	Prep	Ready	Download	Resend	Complete	-	-	-	Finish	Install
AP	02:57	00:43	05:14	00:32	07:13	-	-	-	00:43	01:35
Router	03:04	00:43	05:14	00:32	00:33	05:14	00:32	00:54	00:33	01:33
ED	09:33	-	-	-	00:43	05:14	00:32	00:43	00:43	01:37

While the neighbors of the AP go to the Ready state as soon as the download is completed, other nodes remain in the Sleep state until their respective OTA Masters in the same neighboring node go to the Ready state. The time neighbors of the AP spend in the Ready state is approximately 27 seconds. It takes approximately 3 minutes and 20 seconds to transmit all fragment blocks during the download phase. After the Download, the nodes enter the Resend state (about 21 seconds). After the Resend, the AP's neighbors enter the Complete state and wait for all neighbors' transmissions to be completed (takes 4 minutes and 35 seconds), while the router and its neighbors enter the Ready state and prepare for firmware transmission. After all neighbors' transmissions are completed, the nodes enter the Finish state. Upon completion of the Finish state, all nodes in the network are reset, and the bootloader copies the received firmware binary to the program memory (this process takes approximately 1 minute and 38 seconds).

The time taken for each FOTA step is summarized in the Table 4.

### 6.3. Overall FOTA update time measurement

This test is performed to determine the effect of the number of neighborhoods ( $R$ ) or data length on the total FOTA time. In a lab environment, networks were configured with 1 to 4 neighborhoods were configured, and as in the previous test, only one wireless node was connected in each neighborhood of the network for simplicity.

We conducted experiments with various file sizes, ranging from 1 byte to 100 KB, using the following specific values: 1, 10, 100, 1000, 5000, 10000, 20000, 50000, and 100000 bytes. The total OTA time was recorded for each configuration.

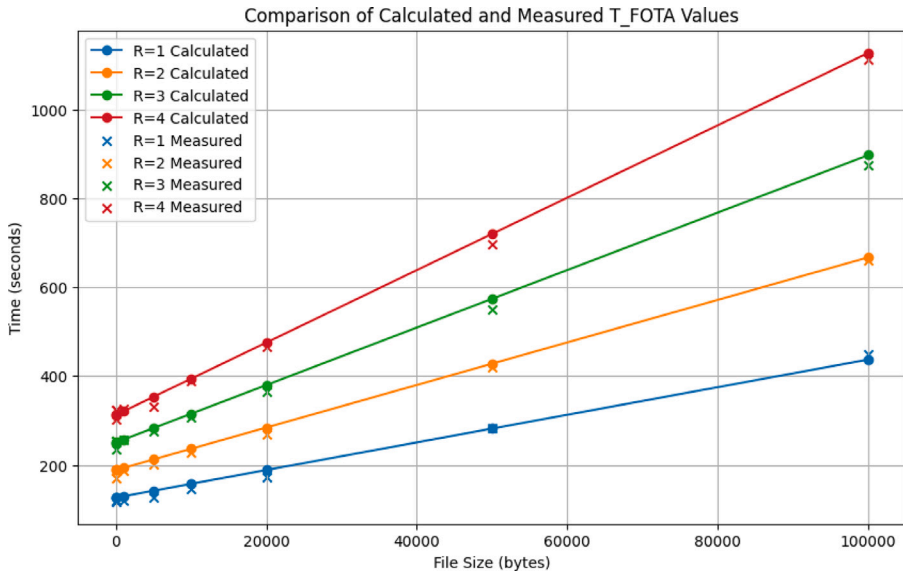


Fig. 15. Calculated and measured  $T_{FOTA}$  values.

Table 5  
Measured  $T_{FOTA}$  values.

Firmware size	Number of neighbors			
	$R = 1$	$R = 2$	$R = 3$	$R = 4$
1	118.0 s	169.0 s	235.0 s	303.5 s
10	118.5 s	187.0 s	255.0 s	324.0 s
100	119.0 s	187.5 s	255.5 s	325.0 s
1,000	119.5 s	188.0 s	256.0 s	325.5 s
5,000	126.0 s	200.5 s	276.5 s	331.5 s
10,000	147.0 s	228.0 s	307.5 s	388.5 s
20,000	173.5 s	269.5 s	366.0 s	466.0 s
50,000	283.5 s	421.0 s	550.0 s	695.5 s
100,000	449.5 s	661.5 s	874.5 s	1113.0 s

In the case of 1 byte files, the purpose is to determine the baseline of the  $T_{Overhead}$  time. For file sizes less than 100 bytes, we illustrate scenarios where only minimal network configuration is transmitted. File sizes less than 5 KB can be considered as cases where only the application layer is being updated [26].

The  $T_{Install}$  phase, which occurs after the download is complete and involves restarting and copying the firmware to flash memory, was excluded from the measurements. Measurements were repeated three times and averaged. Only cases with a BI (Beacon Interval) of 10.81 seconds were measured.

In this experiment, we wanted to correlate the experimental results with the analytical model from Eq. (17) for comparison. Fig. 15 shows the measured  $T_{FOTA}$  values as well as the results calculated using the Algorithm 2, with changes in the firmware binary size and various values of  $R$  for reference.

The graph shows a linear increase with respect to the growth of the  $R$  value and the increase in firmware size. When compared to the values calculated by the Algorithm 2, the observed values show only minor differences.

For file sizes below 100 bytes, however,  $T_{FOTA}$  time remains relatively unchanged with respect to variations in file size, but experiences changes only with increases in the  $R$  value.

The Tables 5 and 6 show the measured and calculated  $T_{FOTA}$  values for varying file sizes and  $R$  values. The measured values represent actual experimental results, while the calculated values are derived from the Algorithm 2.

Overall, these results indicate that the influence of both the number of neighborhoods ( $R$ ) and the firmware binary size have a linear effect on the  $T_{FOTA}$  values. The experimental values show a good agreement with the analytical model. Therefore, a DSME-based network using the proposed FOTA framework can benefit from the overall FOTA time prediction of this paper, allowing an assessment of whether it meets the performance limits required in the application.

#### 6.4. Discussion

In our study, we investigated the OTA (Over-The-Air) performance of a LoRa-based DSME (Distributed Synchronous Multi-channel Extension) network using specific configuration parameters. It is important to note that LoRa modulation parameters, such

**Table 6**  
Calculated  $T_{\text{FOTA}}$  values using Algorithm 2.

Firmware size	Number of neighbors			
	$R = 1$	$R = 2$	$R = 3$	$R = 4$
1	126.8 s	189.1 s	251.3 s	313.6 s
10	126.8 s	189.1 s	251.4 s	313.6 s
100	127.1 s	189.4 s	251.8 s	314.2 s
1,000	129.8 s	193.7 s	257.6 s	321.5 s
5,000	142.3 s	213.0 s	283.6 s	354.3 s
10,000	157.9 s	236.9 s	315.9 s	395.0 s
20,000	188.9 s	284.7 s	380.5 s	476.4 s
50,000	282.2 s	428.4 s	574.7 s	720.9 s
100,000	437.6 s	667.8 s	898.0 s	1128.2 s

as spreading factor (SF), bandwidth (BW), and coding rate (CR), play a crucial role in determining communication range, data rate, and sensitivity to environmental factors.

The choice of SF directly impacts the achievable data rate and range of LoRa transmissions. Lower SF values (e.g., SF = 5) result in higher data rates but shorter communication range, while higher SF values (e.g., SF = 12) offer longer range at the expense of lower data rates. Similarly, the selection of BW influences the channel bandwidth and spectral efficiency, affecting overall communication performance.

Furthermore, the DSME network parameters, including BO (backoff order), MO (multi-superframe order), SO (superframe order), and the slot duration (Symbol Duration), significantly influence the efficiency and reliability of data delivery in multi-hop LoRa networks.

Our results demonstrate the interplay between these parameters and their impact on OTA performance metrics, such as the total time required to deliver firmware updates to all network nodes. The topology of the network, including the number of hops and beacon interval, also affects OTA performance and network scalability.

Future studies could explore the optimization of LoRa and DSME parameters to maximize OTA efficiency and reliability in industrial IoT deployments. Adaptive parameter tuning based on environmental conditions and network dynamics could further enhance the resilience and performance of LoRa-based DSME networks in challenging operational environments.

## 7. Conclusions and future work

This paper presents a comprehensive solution for large-scale IoT networks, designed to enable Firmware Over-the-Air (FOTA) updates in Distributed Synchronous Multi-channel Extension (DSME)-based multi-hop networks, called DSME-FOTA. The study introduces a versatile IoT network design that incorporates the LoRa physical layer (PHY) and a proprietary implementation of the IEEE 802.15.4 DSME MAC.

Furthermore, the design details of the proposed DSME-FOTA framework are presented. It employs a sequential cascade of master-slave operations starting at the Access Point (AP) and extending through multiple routers' neighborhoods. Additionally, a comprehensive quantitative analysis of the performance is presented, with a particular emphasis on the FOTA update time.

The analytical results are compared with our experimental data, and the results show that the FOTA time has a first-order dependence on the number of neighborhoods and the firmware binary size. These quantitative insights offer valuable guidance for understanding the effectiveness and predicting FOTA update times in various network configurations. This research opens up opportunities for further improvements and optimizations in the field of industrial IoT (IIoT) communications.

While our study provides valuable insights into DSME-based multi-hop networking, it is important to acknowledge certain limitations. The experimental setup employed in this study focused on measuring key time values and assessing network performance under controlled laboratory conditions. The simplified testbed environment may not fully capture the complexity and variability present in real-world IIoT deployments, where factors such as dynamic signal conditions and environmental interference could significantly impact network performance. Additionally, the scope of our experiments primarily emphasized basic metrics such as download time and success probability, without exploring more nuanced aspects such as scalability under diverse operating conditions.

As future work, additional field tests need to be conducted to better understand and evaluate the performance of the DSME-FOTA proposed in this paper under real-world conditions with different environmental factors that strongly affect IIoT communications. In addition, FOTA updates will require additional efforts to improve transmission efficiency while maintaining the superframe structure defined by the current DSME standard. Furthermore, future studies may include the implementation of a download pipelining method, which has the potential to lower the overall FOTA time but may involve increased complexity. The feasibility of the protocol also needs to be verified in terms of energy consumption [46] when the wireless nodes are battery-powered.

### CRedit authorship contribution statement

**Jabeom Gu:** Conceptualization, Data curation, Formal analysis, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Seung-Sik Lee:** Conceptualization, Resources, Software, Writing – original draft. **Hoyong Kang:** Conceptualization, Funding acquisition, Project administration, Resources.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

Data will be made available on request.

## Acknowledgments

This work was supported by the Korea Institute of Energy Technology Evaluation and Planning (KETEP) and the Ministry of Trade, Industry & Energy (MOTIE) of the Republic of Korea (No. 20224B1010090).

## References

- [1] W.A. Jabbar, T.K. Kian, R.M. Ramli, S.N. Zubir, N.S.M. Zamrizaman, M. Balfaqih, V. Shepelev, S. Alharbi, Design and fabrication of smart home with Internet of Things enabled automation system, *IEEE Access* 7 (2019) 144059–144074, <http://dx.doi.org/10.1109/ACCESS.2019.2942846>.
- [2] E. Ahmed, I. Yaqoob, A. Gani, M. Imran, M. Guizani, Internet-of-things-based smart environments: State of the art, taxonomy, and open research challenges, *IEEE Wirel. Commun.* 23 (5) (2016) 10–16.
- [3] Y. Kabalci, A survey on smart metering and smart grid communication, *Renew. Sustain. Energy Rev.* 57 (2016) 302–318.
- [4] J. Santa, R. Sanchez-Iborra, P. Rodriguez-Rey, L. Bernal-Escobedo, A.F. Skarmeta, LPWAN-based vehicular monitoring platform with a generic IP network interface, *Sensors* 19 (2) (2019) 264.
- [5] C. Brewster, I. Roussaki, N. Kalatzis, K. Doolin, K. Ellis, IoT in agriculture: Designing a Europe-wide large-scale pilot, *IEEE Commun. Mag.* 55 (9) (2017) 26–33, <http://dx.doi.org/10.1109/MCOM.2017.1600528>.
- [6] N. Nurelmadina, M.K. Hasan, I. Memon, R.A. Saeed, K.A. Zainol Ariffin, E.S. Ali, R.A. Mokhtar, S. Islam, E. Hossain, M.A. Hassan, A systematic review on cognitive radio in low power wide area network for industrial IoT applications, *Sustainability* 13 (1) (2021) 338.
- [7] R.S. Sinha, Y. Wei, S.-H. Hwang, A survey on LPWA technology: LoRa and NB-IoT, *ICT Express* 3 (1) (2017) 14–21, <http://dx.doi.org/10.1016/j.icte.2017.03.004>.
- [8] A. Ikpehai, B. Adebisi, K.M. Rabie, K. Anoh, R.E. Ande, M. Hammoudeh, H. Gacanin, U.M. Mbanaso, Low-power wide area network technologies for Internet-of-Things: A comparative review, *IEEE Internet Things J.* 6 (2) (2019) 2225–2240, <http://dx.doi.org/10.1109/JIOT.2018.2883728>.
- [9] B. Buurman, J. Kamruzzaman, G. Karmakar, S. Islam, Low-power wide-area networks: Design goals, architecture, suitability to use cases and research challenges, *IEEE Access* 8 (2020) 17179–17220, <http://dx.doi.org/10.1109/ACCESS.2020.2968057>.
- [10] L.D. Xu, W. He, S. Li, Internet of Things in industries: A survey, *IEEE Trans. Ind. Inform.* 10 (4) (2014) 2233–2243, <http://dx.doi.org/10.1109/TII.2014.2300753>.
- [11] L. Shu, M. Mukherjee, M. Pecht, N. Crespi, S.N. Han, Challenges and research issues of data management in IoT for large-scale petrochemical plants, *IEEE Syst. J.* 12 (3) (2018) 2509–2523, <http://dx.doi.org/10.1109/JSYST.2017.2700268>.
- [12] G. Bedi, G.K. Venayagamoorthy, R. Singh, R.R. Brooks, K.-C. Wang, Review of Internet of Things (IoT) in electric power and energy systems, *IEEE Internet Things J.* 5 (2) (2018) 847–870, <http://dx.doi.org/10.1109/JIOT.2018.2802704>.
- [13] D.J. Naus, Inspection of Nuclear Power Plant Structures - Overview of Methods and Related Applications, Tech. Rep. ORNL/TM-2007/191, 2009, <http://dx.doi.org/10.2172/969948>.
- [14] W. Tu, L. Meng, Q. Ye, M. Shen, Y. Xu, Propagation characteristics of lora-based wireless communication in steel ship cabin, in: S. Paiva, X. Li, i.I. Lopes, N. Gupta, D.B. Rawat, A. Patel, H.R. Karimi (Eds.), *Science and Technologies for Smart Cities*, Vol. 442, Springer International Publishing, Cham, 2022, pp. 647–658.
- [15] W. Tu, H. Xu, Y. Xu, Q. Ye, M. Shen, Research on 2.4 GHz wireless channel propagation characteristics in a steel ship cabin, in: A. Mase (Ed.), *Int. J. Antennas Propag.* 2021 (2021) 1–12, <http://dx.doi.org/10.1155/2021/6623638>.
- [16] R. Perveen, N. Kishor, S.R. Mohanty, Off-shore wind farm development: Present status and challenges, *Renew. Sustain. Energy Rev.* 29 (2014) 780–792, <http://dx.doi.org/10.1016/j.rser.2013.08.108>.
- [17] V.C. Gungor, G.P. Hancke, Industrial wireless sensor networks: Challenges, design principles, and technical approaches, *IEEE Trans. Ind. Electron.* 56 (10) (2009) 4258–4265, <http://dx.doi.org/10.1109/TIE.2009.2015754>.
- [18] A. Nikoukar, S. Raza, A. Poole, M. Güneş, B. Dezfouli, Low-power wireless for the Internet of Things: standards and applications, *IEEE Access* 6 (2018) 67893–67926, <http://dx.doi.org/10.1109/ACCESS.2018.2879189>.
- [19] A. Balador, A. Kouba, D. Cassioli, F. Foukalas, R. Severino, D. Stepanova, G. Agosta, J. Xie, L. Pomante, M. Mongelli, P. Pierini, S. Petersen, T. Sukuvaara, Wireless communication technologies for safe cooperative cyber physical systems, *Sensors* 18 (11) (2018) 4075, <http://dx.doi.org/10.3390/s18114075>.
- [20] H. Kurunathan, R. Severino, A. Koubaa, E. Tovar, IEEE 802.15.4e in a Nutshell: Survey and performance evaluation, *IEEE Commun. Surv. Tutor.* 20 (3) (2018) 1989–2010, <http://dx.doi.org/10.1109/COMST.2018.2800898>.
- [21] IEEE standard for local and metropolitan area networks—Part 15.4: Low-rate wireless personal area networks (LR-WPANs) amendment 1: MAC sublayer, in: *IEEE Std 802.15.4e-2012 (Amendment to IEEE Std 802.15.4-2011)*, 2012, pp. 1–225, <http://dx.doi.org/10.1109/IEEESTD.2012.6185525>.
- [22] J.R. Cotrim, J.H. Kleinschmidt, LoRaWAN mesh networks: A review and classification of multihop communication, *Sensors* 20 (15) (2020) 4273, <http://dx.doi.org/10.3390/s20154273>.
- [23] M. Taneja, A framework to support real-time applications over IEEE802.15.4 DSME, in: 2015 IEEE Tenth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2015, pp. 1–6, <http://dx.doi.org/10.1109/ISSNIP.2015.7106918>.
- [24] K. Ray, S. Moulik, I-DSME: An industrial-DSME MAC protocol for smart factory automation, *Internet of Things* 23 (2023) 100859, <http://dx.doi.org/10.1016/j.iot.2023.100859>.
- [25] K. Zandberg, K. Schleiser, F. Acosta, H. Tschofenig, E. Baccelli, Secure firmware updates for constrained IoT devices using open standards: A reality check, *IEEE Access* 7 (2019) 71907–71920, <http://dx.doi.org/10.1109/ACCESS.2019.2919760>.
- [26] J. Bauwens, P. Ruckebusch, S. Giannoulis, I. Moerman, E.D. Poorter, Over-the-air software updates in the Internet of Things: An overview of key principles, *IEEE Commun. Mag.* 58 (2) (2020) 35–41, <http://dx.doi.org/10.1109/MCOM.001.1900125>.
- [27] S. El Jaouhari, E. Bouvet, Secure firmware over-the-air updates for IoT: Survey, challenges, and discussions, *Internet of Things* 18 (2022) 100508, <http://dx.doi.org/10.1016/j.iot.2022.100508>.
- [28] E. Nett, WLAN in automation—more than an academic exercise? in: *Latin-American Symposium on Dependable Computing*, Springer, 2005, pp. 4–8.

- [29] A. Willig, K. Matheus, A. Wolisz, Wireless technology in industrial networks, *Proc. IEEE* 93 (6) (2005) 1130–1151, <http://dx.doi.org/10.1109/JPROC.2005.849717>.
- [30] A.W. Al-Dabbagh, T. Chen, Design considerations for wireless networked control systems, *IEEE Trans. Ind. Electron.* 63 (9) (2016) 5547–5557, <http://dx.doi.org/10.1109/TIE.2016.2564950>.
- [31] M. Luvisotto, F. Tramarin, L. Vangelista, S. Vitturi, On the use of LoRaWAN for indoor industrial IoT applications, *Wirel. Commun. Mob. Comput.* 2018 (2018) 1–11, <http://dx.doi.org/10.1155/2018/3982646>.
- [32] Y.-H. Wei, Q. Leng, S. Han, A.K. Mok, W. Zhang, M. Tomizuka, RT-WiFi: Real-time high-speed communication protocol for wireless cyber-physical control applications, in: 2013 IEEE 34th Real-Time Systems Symposium, IEEE, Vancouver, BC, Canada, 2013, pp. 140–149, <http://dx.doi.org/10.1109/RTSS.2013.22>.
- [33] P. Danielis, J. Skodzik, V. Altmann, E.B. Schweissguth, F. Golasowski, D. Timmermann, J. Schacht, Survey on real-time communication via ethernet in industrial automation environments, in: Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA), 2014, pp. 1–8, <http://dx.doi.org/10.1109/ETFA.2014.7005074>.
- [34] S. Vitturi, C. Zunino, T. Sauter, Industrial communication systems and their future challenges: Next-generation ethernet, IIoT, and 5G, *Proc. IEEE* 107 (6) (2019) 944–961, <http://dx.doi.org/10.1109/JPROC.2019.2913443>.
- [35] R.A. Swartz, J.P. Lynch, S. Zerbst, B. Sweetman, R. Rolfes, Structural monitoring of wind turbines using wireless sensor networks, *Smart Struct. Syst.* 6 (3) (2010) 183–196, <http://dx.doi.org/10.12989/SSS.2010.6.3.183>.
- [36] M.L. Adekanbi, Optimization and digitization of wind farms using Internet of Things: A review, *Int. J. Energy Res.* 45 (11) (2021) 15832–15838, <http://dx.doi.org/10.1002/er.6942>.
- [37] H. Kdouh, G. Zaharia, C. Brousseau, H. Farhat, G. Grunfelder, G. El, Application of wireless sensor network for the monitoring systems of vessels, in: M. Matin (Ed.), *Wireless Sensor Networks - Technology and Applications*, InTech, 2012, pp. 285–308, <http://dx.doi.org/10.5772/48276>.
- [38] Q. Huang, J. Jiang, Y.Q. Deng, Comparative evaluation of three wireless sensor network transceivers in a high radiation environment, *EPJ Web Conf.* 225 (2020) 08007, <http://dx.doi.org/10.1051/epjconf/202022508007>.
- [39] S. Brown, C. Sreenan, *Updating Software in Wireless Sensor Networks: A Survey*, Tech. Rep., Dept. of Computer Science, National Univ. of Ireland, Maynooth, 2006, pp. 1–14.
- [40] S. Brown, C.J. Sreenan, Software updating in wireless sensor networks: A survey and lacunae, *J. Sens. Actuator Netw.* 2 (4) (2013) 717–760, <http://dx.doi.org/10.3390/jsan2040717>.
- [41] J.W. Hui, D. Culler, The dynamic behavior of a data dissemination protocol for network programming at scale, in: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, 2004, pp. 81–94.
- [42] K. Abdelfadeel, T. Farrell, D. McDonald, D. Pesch, How to make firmware updates over LoRaWAN possible, in: 2020 IEEE 21st International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), pages=16–25, doi=10.1109/WoWMoM49955.2020.00018,, 2020.
- [43] T. Stathopoulos, J. Heidemann, D. Estrin, A remote code update mechanism for wireless sensor networks, Tech. rep., CENS-TR-30, University of California, LA, 2003.
- [44] K. Arakadakis, P. Charalampidis, A. Makrogiannakis, A. Fragkiadakis, Firmware over-the-air programming techniques for IoT networks - A survey, *ACM Comput. Surv.* 54 (9) (2021) 178:1–178:36, <http://dx.doi.org/10.1145/3472292>.
- [45] S. Kulkarni, L. Wang, Energy-efficient multihop reprogramming for sensor networks, *ACM Trans. Sensor Netw.* 5 (2) (2009-04-03) 16:1–16:40, <http://dx.doi.org/10.1145/1498915.1498922>.
- [46] P. Ruckebusch, S. Giannoulis, I. Moerman, J. Hoebeke, E. De Poorter, Modelling the energy consumption for over-the-air software updates in LPWAN networks: SigFox, LoRa and IEEE 802.15.4g, *Internet of Things* 3–4 (2018) 104–119, <http://dx.doi.org/10.1016/j.iot.2018.09.010>.
- [47] B. Moran, H. Tschofenig, D. Brown, M. Meriac, A firmware update architecture for Internet of Things, Request for Comments RFC 9019, Internet Engineering Task Force, 2021, <http://dx.doi.org/10.17487/RFC9019>.
- [48] M.J.B. de Sousa, L.F.G. Gonzalez, E.M. Ferdinando, J.F. Borin, Over-the-air firmware update for IoT devices on the wild, *Internet of Things* 19 (2022) 100578, <http://dx.doi.org/10.1016/j.iot.2022.100578>.
- [49] F. Mahfoudhi, A.K. Sultania, J. Famaey, Over-the-air firmware updates for constrained NB-IoT devices, *Sensors* 22 (19) (2022) 7572, <http://dx.doi.org/10.3390/s22197572>.
- [50] H. Chandra, E. Anggadajaja, P.S. Wijaya, E. Gunawan, Internet of Things: Over-the-air (OTA) firmware update in lightweight mesh network protocol for smart urban development, in: 2016 22nd Asia-Pacific Conference on Communications (APCC), 2016, pp. 115–118, <http://dx.doi.org/10.1109/APCC.2016.7581459>.
- [51] C. Zhang, W. Ahn, Y. Zhang, B.R. Childers, Live code update for IoT devices in energy harvesting environments, in: 2016 5th Non-Volatile Memory Systems and Applications Symposium (NVMSA), 2016, pp. 1–6, <http://dx.doi.org/10.1109/NVMSA.2016.7547182>.
- [52] J. Álamos, P. Kietzmann, T.C. Schmidt, M. Wählisch, DSME-LoRa: Seamless long-range communication between arbitrary nodes in the constrained IoT, *ACM Trans. Sensor Netw.* 18 (4) (2022) 1–43, <http://dx.doi.org/10.1145/3552432>.
- [53] F. Cuomo, S. Della Luna, E. Cipollone, P. Todorova, T. Suihko, Topology formation in IEEE 802.15.4: Cluster-tree characterization, in: 2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom), 2008, pp. 276–281, <http://dx.doi.org/10.1109/PERCOM.2008.26>.
- [54] Q. Wang, Y. Zhu, L. Cheng, Reprogramming wireless sensor networks: Challenges and approaches, *IEEE Netw.* 20 (3) (2006) 48–55, <http://dx.doi.org/10.1109/MNET.2006.1637932>.
- [55] D. Heeger, M. Garigan, E. Eleni Tsiropoulou, J. Plusquellic, Secure LoRa firmware update with adaptive data rate techniques, *Sensors* 21 (7) (2021) 2384, <http://dx.doi.org/10.3390/s21072384>.