IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received May 16, 2020, accepted June 11, 2020, date of publication July 8, 2020, date of current version July 21, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3007985

A Two-Hop Real-Time LoRa Protocol for Industrial Monitoring and Control Systems

HUU PHI TRAN[1], WOO-SUNG JUNG[2], (Member, IEEE), TAEHYUN YOON[2], DAE-SEUNG YOO[2], AND HOON OH[1], (Member, IEEE)

[1]Department of Electrical and Computer Engineering, University of Ulsan, Ulsan 680-749, South Korea
[2]Electronics and Telecommunications Research Institute, Daejeon 305-350, South Korea

Corresponding author: Hoon Oh (hoonoh@ulsan.ac.kr)

This work was supported by the Electronics and Telecommunications Research Institute (ETRI) and by a grant funded by the Korean Government and Ulsan Metropolitan City (20ZS1200, The development of a smart context-awareness foundation technique for major industry acceleration; 20AS1100, Development of a smart HSE system and a digital cockpit system based on ICT convergence for enhanced major industry).

ABSTRACT A single-hop LoRa (**Lo**ng **Ra**nge) network suffers from a limitation in coverage caused by high signal attenuation when deployed in industrial areas, which often include wireless unfriendly zones (WUZs). Thus, some end nodes in WUZs may fail to transmit data to the gateway (GW). This paper proposes a reliable two-hop real-time LoRa protocol in which the nodes in WUZs transmit data to another node that relays the received data to the gateway for reliable transmission. The proposed protocol allocates transmission slots to nodes with different data transmission periods, so the nodes not only achieve high reliability but also meet time constraints or periods for delivering data to a gateway. In this case, 1-hop nodes (one hop away from GW) may consume more energy than 2-hop nodes (two hops away from GW). For energy balancing, the protocol uses a data aggregation approach to minimize the number of transmissions. We show by analysis that the proposed protocol can support up to 200 nodes using a single-channel GW when every node sends one packet with a payload of 30 bytes within 17 seconds, and 30% of the nodes are 2-hop nodes. In addition, we show by experiment that the proposed protocol can achieve high reliability in data transmission.

INDEX TERMS Two-hop LoRa network, energy balancing, relay, real-time scheduling, reliable transmission.

I. INTRODUCTION

Recently, LoRa (long range) technology, despite its low spectrum efficiency and low data rate, has been drawing attention as an alternative or a complement to wireless sensor networks (WSNs) owing to its provision of long-range communications and a reliable link that enables a simple star topology. However, the use of a LoRa network in industrial monitoring and control applications is still a challenge since the network has to cover *wireless-unfriendly zones* (WUZs) such as underground tunnels, enclosed spaces, and harsh construction houses with steel or concrete obstructions, but still has to deliver data in a reliable and real-time manner. Furthermore, those applications deal with a relatively high data rate that can increase the possibility of collision. Therefore, it is necessary to design an efficient multi-hop LoRa protocol that can satisfy these industry requirements.

Basically, LoRa technology can overcome the effect of signal attenuation to some extent by increasing the coverage of the network with the use of higher spreading factors (SFs). However, an increase in the SF by 1 degrades the data rate by the power of 2. Much worse, it aggravates the problem of data collisions due to the lengthier time on air (ToA) and the longer transmission range. One well-known LoRa protocol, the LoRa wide area network (LoRaWAN), operates based on the star network topology [1]–[3]. This approach employs the concept of the Aloha protocol [4] in which an end device (node) transmits data freely as long as it has data. Once a node sends data to a server via a gateway (GW), it waits for a command or message from the server in the next two downlink slots. This protocol targets the applications with short duty cycles or long data transmission intervals due to the high possibility of a collision. According to previous

The associate editor coordinating the review of this manuscript and approving it for publication was Mingjun Dai.

studies, it is known that the rate of successfully transmitted packets decreases significantly as the number of end nodes increases [5]–[9].

A number of studies have been conducted to overcome the reliability problems in LoRaWAN [10]–[15]. In [10], the authors proposed two-step lightweight scheduling that helps a node determine its transmission parameters, such as transmission power, spreading factor, and channel, thereby improving the reliability and scalability of the LoRaWAN. However, the protocol still suffers from high packet error rates with the increase in traffic load. In [11] and [12], the authors examined the applicability of time-slotted channel hopping (TSCH) [16] to a LoRaWAN using a small LoRa network of one GW and three nodes. However, such a modification did not achieve notable improvement over the legacy LoRaWAN. In the on-demand LoRa protocol [13], each node includes a wake-up radio that enables a low-energy listening state. The GW sends a beacon message to a cluster head that is responsible for scheduling the data transmissions of its member(s). This approach improves reliability by removing collisions within a cluster; however, the cluster head may suffer from high energy consumption, since it has to remain in listening mode. Meanwhile, Symphony Link [14] employs the notion of slotted Aloha and the listen-before-talk (LBT) mechanism to reduce collisions. It allocates a sufficient number of slots to only high-priority packets; the nodes with low-priority packets send data in a channel contention manner. In [15], the authors proposed a low-overhead synchronization and scheduling mechanism for class A LoRaWAN devices. Data transmissions are scheduled in time slots. It was verified by simulation that the slot-scheduled approach could greatly improve the packet delivery rate.

The above protocols using a star topology have limitations in covering WUZs. Furthermore, if the line of sight (LOS) is not secured, the signal attenuation becomes worse [17], [18]. In order to deal with this problem, some researchers have proposed multi-hop LoRa protocols [19]–[22]. In [19] and [20], the authors proposed Glossy-based [23] multi-hop LoRa protocols that take advantage of the capability for concurrent transmission (CT) such that a node can demodulate one message correctly if two or more messages are transmitted within three symbol periods. In LoRaBlink [19], the protocol repeats a time frame that consists of a network construction (NC) period and a data transmission (DT) period, both being logically divided into a number of slots. In the NC period, the sink floods the network with a beacon message to construct the network, in which every node knows the hop count from the sink. In the DT period, a node that has data to send selects the next available slot and broadcasts the data. Upon receiving data, every node that is located at one level lower than the sender rebroadcasts the data in the slot immediately following, so that the receiver can demodulate the data correctly. This process allows multiple nodes to transmit their own data simultaneously, and the receiver can demodulate the transmission with the strongest signal while it suppresses the other signals. In addition, the process can consume high amounts of energy due to the nature of flooding. In the concurrent transmission LoRa (CT-LoRa) protocol [20], the authors dealt with how a multi-hop node can send data reliably to a sink by making use of CT. Instead of using network construction (like the previous approach), a source broadcasts data freely within the assigned slot. Upon receiving data, a node assigns a time offset before rebroadcasting, so that the receiver can better demodulate the data. Basically, this approach improves the concept of the Glossy approach by utilizing the characteristics of LoRa technology. In a typical multiple-building area network (MBAN) using 18 end nodes, CT-LoRa could achieve a packet delivery rate of almost 100%. However, if many nodes are involved in transmitting data, it becomes difficult to assign appropriate offsets. Furthermore, flooding-based transmission is not free from the efficiency of energy management. In [21], the authors proposed the LoRa-Mesh protocol in which the GW maintains a whole tree topology for the participating nodes by receiving the link state from every node, and sends a query message to a specified node to get data along a tree path. This approach could improve the packet delivery rate significantly, compared to LoRaWAN in network scenarios deployed in both campus and indoor environments. However, it causes high overhead, since it has to query the node whenever it wants to get data from the node. The authors in [22] proposed the synchronous LoRa-Mesh (Sync-LoRa-Mesh) protocol to collect data reliably from WUZs like underground or indoor areas. It defines a repeater node (RN) that has reliable LoRaWAN connectivity to the GW. The RN forms a tree with underground nodes, and collects data in a synchronous manner through scheduling, and it then transmits aggregated data to the GW using the LoRaWAN. It could improve the packet error rate significantly for tree nodes. However, the RN still has to rely on the LoRaWAN, which is not suitable for high-traffic monitoring systems.

In this paper, a new two-hop real-time LoRa protocol is proposed that enables end nodes to transmit data reliably to a gateway in industrial fields including WUZs. The protocol first constructs a two-hop tree topology by taking into account the link quality between end nodes and the gateway. Based on the two-hop tree, a server performs slot scheduling. Then, the nodes at tree level 1 (*1-hop nodes*) can transmit data directly to the GW, while the nodes at tree level 2 (*2-hop nodes*) can transmit data via another 1-hop node to the GW without collision by using the scheduled slots. In this approach, two important things have to be achieved. One is to construct a reliable two-hop tree autonomously, and another is to perform slot scheduling efficiently such that every node, either 1-hop or 2-hop, with different data transmission periods should be able to send data to the GW within the specified period or by the deadline.

The rest of this paper is organized as follows. Section II gives the background to this research, and Section III describes the proposed protocol formally, followed by evaluation of the protocol in Section IV. Concluding remarks are given in Section V.

II. BACKGROUND

A. NETWORK MODEL

The considered LoRa network consists of a network server (NS), multiple gateways, and a number of end devices (end nodes or nodes) where the NS and GWs are interconnected by a high-speed backhaul network. The NS collects data from many participating nodes through the GWs, and provides the necessary services based on analysis of the collected data. Since one GW has to deal with many nodes, in general, it is wall-powered, whereas end nodes are battery-powered.

Every node is required to send data to the server during its own transmission period, specified during network initialization. Therefore, every node can have a different data transmission period. Even though a wireless link in the LoRa network has a sufficiently long range, if a node is deployed in a WUZ, it may not directly connect to the GW in a reliable manner. Thus, nodes form a two-hop tree topology originating from the GW, where each tree consists of *1-hop nodes* that can connect to the GW directly, and zero or more *2-hop nodes* that can connect to the GW via an 1-hop node or a *relay node*. Then, the 2-hop node needs a relay node that can forward its data to the GW. Every node can act as a relay node. It is believed that a two-hop tree is sufficient to cover large industry fields, since two GWs can cover a range that corresponds to five wireless hops. A node that belongs to a tree is said to be a *tree-node*, and a link between that node and its parent is a *tree-link*.

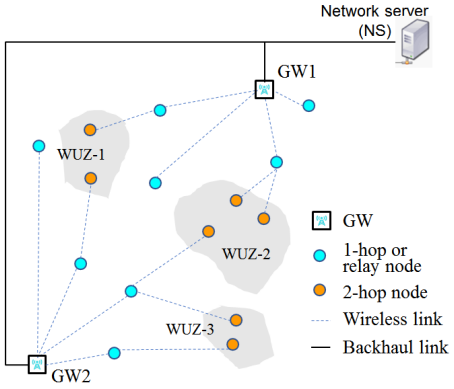


FIGURE 1. Network topology.

Fig. 1 shows one typical two-hop network that can be deployed in industrial areas. The network consists of two GWs and 15 end nodes where eight 1-hop nodes (cyan circles) are located in open spaces, and seven 2-hop nodes (brown circles) are located in WUZs. Once each tree is constructed, the proposed protocol works in the same way for different trees. Therefore, in the rest of this paper, the protocol design considered will be for only one tree.

B. NOTATIONS AND MESSAGES

Some notations and messages are defined for convenience in explanations. A data collection cycle, or a *frame*, is divided into a number of *time slots* (*slots*) such that the length of a slot is sufficient to send one data packet. Since each node is required to send data, it can be modeled as a *data transmission task* (a *task*). Tasks can belong to different classes according to their data transmission period, such that for a frame with 2^N slots where N is an integer value greater than 0, if a task has as its transmission period $P = 2^N/2^c$, this task belongs to class $c (0 \leq c \leq N)$. For example, a class 0 task ($c = 0$) has to transmit one data segment per frame. Then, a task of node x, denoted by τ_x, can be represented as a *task profile* that includes its node address and class:

$$\tau_x = (x, class(x))$$

where x and $class(x)$ indicate the address and the class of node x, respectively.

Some of the messages that are used to build a two-hop tree are defined as follows.

Message	Description
JREQ = (*forced-flag*)	A node sends a *join request* (JREQ) to a tree-node in order to become a child. If *forced-flag* = 1, the receiving node must accept the sender as a child, regardless of link quality; otherwise, it may reject the join request based on the number of its children.
JRES = (*accept-flag*)	A tree-node sends a *join response* (JRES) to the JREQ. It responds with *accept-flag* = 1 to indicate that it accepts the join request.
TCR = (*nd, ndaddr, ndlevel*)	Node *nd*, either a tree-node or a GW, broadcasts a *tree construction request* (TCR) to construct a tree topology, where *ndaddr* and *ndlevel* indicate the address and the tree level, respectively, of sending node *nd*.

C. THE LOGICAL SLOT INDEXING ALGORITHM

It is not easy to schedule a set of tasks with different data transmission periods so that every task in the set always completes its data transmission before the time constraint or the end of its period. For convenience in task scheduling, we borrow the notion of *logical slot indexing* (LSI) [24], in which the LSI algorithm assigns a logical slot index to every slot in an array of slots. The importance of the logical slot index is as follows. In a frame with 2^N slots, if a task of class k is assigned 2^k slots corresponding to the 2^k sequential logical slot indices starting with any logical slot index, and it transmits data in each assigned slot, it can always satisfy its time constraint.

To explain the LSI algorithm, we introduce the notion of the 2^k-*Constraint*. It is said that 2^k logical indices satisfy a 2^k-*Constraint* if each of the 2^k logically indexed slots appear

in only one of the 2^k equally divided parts of a frame. The LSI algorithm is given briefly in Algorithm 1.

Algorithm 1 Logical Slot Indexing

// Assume that a frame has 2^N physical slots
1 2^1 logical indices are assigned such that they satisfy the $2^1 - Constraint$;
2 **for** $i = 2$ to N **do**
3 **for** $j = 2^{i-1} + 1$ to 2^i **do**
4 assign j s.t. $\forall k = 1 \ldots i$, the logical indices from j to $j + 2^k - 1$ satisfy the $2^k - Constraint$;

In Algorithm 1, the statement in line 1 guarantees that the first two logical indices are assigned so they appear in each of two equally divided parts. When $i = 2$ (line 2), logical indices 3 and 4 are assigned (line 3) such that when 3 is assigned, (2, 3) satisfies the $2^1 - Constraint$, and when 4 is assigned, (4, 3) satisfies the $2^1 - Constraint$, and (1, 2, 3, 4) also satisfies the $2^2 - Constraint$ (line 4) in a recursive manner. When $i = 3$, the logical indices 5, 6, 7, and 8 are assigned, similarly satisfying the corresponding constraints. A detailed description of the LSI algorithm is in [24].

Consider task scheduling in a frame with 2^3 slots. Fig. 2 shows a frame-slot architecture with the logical slot indices and the slot scheduling of two tasks: τ_A=$(A, 0)$ and τ_B=$(B, 2)$. Task τ_A takes physical slot 1 corresponding to logical slot index 1, and task τ_B takes slots 2, 3, 5, and 7 corresponding to logical slot indices 2, 3, 4, and 5, respectively. Then, tasks A and B transmit one packet per the period of 2^3 slots and the period of 2^1 slots, respectively.

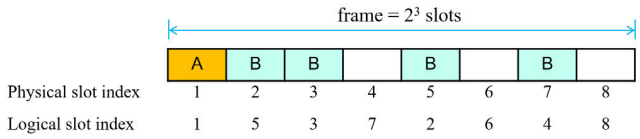


FIGURE 2. A frame-slot architecture of $2^3 2^3$ slots with the logical slot indices assigned and the slot scheduling of two tasks: $\tau_A = (A, 0)$ and $\tau_B = (B, 2)$.

D. MOTIVATION

LoRa technology, characterized by the provision of a long-range and reliable link, enables a star topology LoRa network in which a LoRa GW collects data directly from end nodes against node mobility. The star topology makes it easy to design a data transmission protocol. However, data transmission in the LoRa network suffers from data collisions due to a lengthy packet ToA. Hence, a simple protocol like LoRaWAN is not suitable for industrial monitoring and control systems with relatively high data traffic. Furthermore, the stability of a LoRa link is vulnerable to any failure to secure the LOS. Hence, the GW often fails to cover the nodes deployed in WUZs, thereby requiring a relay node that receives data from those nodes and forwards them to the GW.

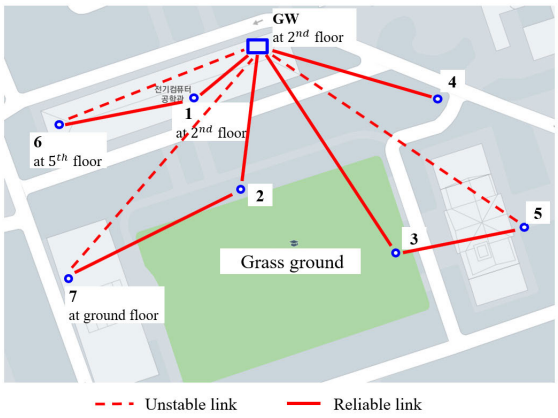


FIGURE 3. A small LoRa network deployed across multiple buildings on the campus of the University of Ulsan.

Experiments were set up in a university campus testbed that included buildings and closed rooms, as shown in Fig. 3, where the GW was placed in the laboratory, three nodes (1, 6, and 7) were inside the buildings, and four nodes (2, 3, 4, and 5) were outside. Every node was programmed to send a packet with a payload of 20 bytes at regular intervals of five seconds with SF = 7 on a single channel for 500 seconds. A comparison of *packet reception ratios* (*PRRs*) was made between nodes 5, 6, and 7 sending data directly to the GW and sending data via nodes 3, 1, and 2, respectively. The results are summarized in Table 1. Two-hop transmission shows clear improvement, whereas node 7 experienced connection failure with the direct one-hop transmission.

TABLE 1. Comparison of PRRs for one-hop and two-hop transmissions in FIG. 3.

Node	5	6	7
One-hop transmission	25%	53%	0%
Two-hop transmission	99%	98%	97%

Meanwhile, some problems were derived from the two-hop LoRa network. *First*, two-hop transmission can increase the possibility of collision, since the data generated by 2-hop nodes have to be transmitted twice, and thus, the increased traffic can increase interference due to the long transmission range. The interference problem is often resolved by employing time-division multiple access (TDMA). *Second*, 2-hop nodes with different transmission periods also have to send data within their respective periods or time constraints. A slot schedule for a two-hop tree requires efficient slot scheduling and schedule distribution. *Third*, a balanced two-hop tree topology has to be constructed to distribute the processing overhead among the 1-hop relay nodes under the constraint that every tree-link should be reliable. *Lastly*, an 1-hop relay node may consume more energy, since it has to forward the received data. In the LoRa technology, a node consumes

much more energy in sending mode [25]. Aggregated data transmission may alleviate this problem.

The above problems are addressed with the following design principles. Upon receiving a tree construction request, every node can evaluate the link quality by examining the received signal strength indicator (RSSI) and the signal-to-noise ratio (SNR), and can then judge whether they are adequate for it to become an 1-hop node. Furthermore, for energy balancing, an 1-hop relay can limit the number of nodes it can handle. Once the two-hop tree is constructed, every node can share its task profile with other nodes via the GW, and can generate an identical slot schedule for all the participating nodes based on the frame-slot architecture and the logical slot indices. A 2-hop node needs two slots within its data transmission period for its own transmission and its parent's relay. This can be done using the logical slot indices. After slot scheduling, the 1-hop node can locally rearrange the transmission sequence for itself and its children and determine which slots are to be used for transmission (or skipped) to maximize data aggregation. In this paper, a two-hop real-time LoRa (Two-hop RT-LoRa) protocol is designed based on the above design principles.

III. THE TWO-HOP REAL-TIME LoRa PROTOCOL

A. PROTOCOL AND FRAME-SLOT STRUCTURE

The protocol structure consists of the *network construction* (NC) period, the *data collection* (DC) period, and an optional *network maintenance* (NM) period. During the NC period, nodes form a two-hop tree originating at GW. Then, during the DC period, GW repeats the DC cycle that corresponds to the frame. After a series of frames, GW may start an optional NM period if broken links need to be fixed.

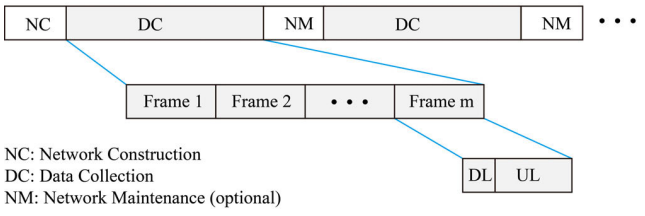


FIGURE 4. Protocol structure.

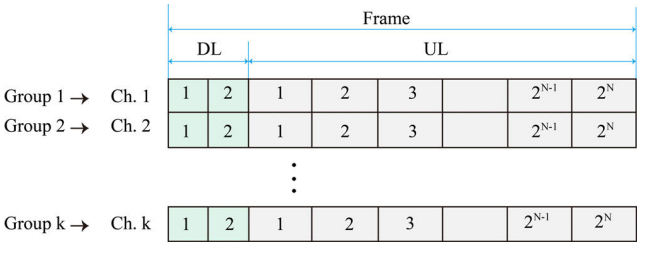


FIGURE 5. Frame-slot architecture.

A frame is divided into a *downlink* (DL) period and an *uplink* (UL) period. The DL period is further divided into two slots, one for transmission of a command to 1-hop nodes, and another for rebroadcasting the command toward 2-hop nodes. At the beginning of the DL period, all nodes listen to the common channel, say Ch. 1, to receive a command from the GW. Upon receiving the command, every node synchronizes the start time of the UL period. The UL period is sliced into a number of data transmission slots. Multiple channels can define multiple identical frames. A slot consists of three parts: guard time, channel activity detection (CAD) time, and transmission (Tx) time. The guard time is used to make up for time synchronization errors. Tx time is also referred to as the packet ToA.

B. TREE CONSTRUCTION AND MAINTENANCE

A tree is constructed such that every node has a good tree-link quality, and an 1-hop relay node will limit the number of children to ensure energy balancing. When a node, say x, receives a TCR from node y, it evaluates the quality of the link (x, y) using evaluation function $linkq(x, y)$:

$$linkq(x,y) = \begin{cases} good \text{ if } RSSI \geq RSSITh \text{ and } SNR \geq SNRTh \\ fair \text{ if } RSSI \geq RSSITh \text{ or } SNR \geq SNRTh \\ bad \text{ if } RSSI < RSSITh \text{ and } SNR < SNRTh \end{cases}$$

where $RSSITh$ and $SNRTh$ are the threshold values for RSSI and SNR, respectively, which indicate the minimum values for good link quality. Every node x maintains a tree information table, $TIT(x)$, as follows:

$$TIT(x) = (GW, P(x), level(x), linkq(x, P(x)), C(x))$$

where GW is the gateway to which node x belongs, $P(x)$ is node x's parent, $level(x)$ is the level of node x, and $C(x)$ indicates the set of node x's children.

Tree construction is performed as follows. The GW broadcasts TCR = (GW, GW, 0) to initiate tree construction. Upon receiving the TCR from y, node x becomes an 1-hop node only if the *ndlevel* in the TCR is zero and $linkq(x, y)$ is good. Then, it updates $TIT(x)$ with $level(x) = level(y) + 1$ and rebroadcasts the TCR after a random delay only if $level(x) = 1$. After the delay, an orphan node, say x, selects a node that provides a good link quality in $TIT(x)$ as a candidate for its parent. Then, it tries to join the selected node, say y, by sending JREQ = (0). Upon receiving the JREQ, node y responds with JRES = (1) only if $|C(y)|$ is less than or equal to the specified maximum limit, *maxChildren*, for energy balancing. If node x receives JRES = (1), it determines node y as its parent. If it does not receive JRES, it launches the join process one more time after a random delay. After two failures, node x executes the join process by excluding the candidate parents that were tried previously. If it again receives JRES = (0), it tries the join process with another parent candidate. In this process, every node x can maintain its $C(x)$. Some remaining orphan nodes can join the 1-hop nodes that provide a fair link quality by sending JREQ = (1). If necessary, some additional GWs can be installed.

If a node does not receive a command from its parent for three consecutive frames, it judges that it has lost its parent.

If an 1-hop node fails to receive data from a specific child for three consecutive frames, it removes the child from its children list and releases the corresponding slot schedule. A disconnected node or new node waits for the next NM period to try the join process. In the NM period, a node informs its child of a bad link if it finds the PRR from the child is less than a specified value, so it can find a new parent.

C. TIME SYNCHRONIZATION

Simple time synchronization is performed, considering the low data rate of LoRa technology. A GW broadcasts a *synchronization* (SYN) message at the start of a frame period, *StartDL*, as illustrated in Fig. 6. Upon receiving SYN, every 1-hop node that has at least one child rebroadcasts it at *StartRB*:

$$StartRB = sysTime - ToA + {}^{DL}/_{2} \tag{1}$$

In this process, if a 2-hop node receives multiple SYNs, it will get one with the strongest signal based on the notion of concurrent transmission [23]. Upon receiving SYN, a node can calculate the start time of the UL period, *startUL*, as follows:

$$StartUL = \begin{cases} sysTime - ToA + DL & \text{for 1-hop node} \\ sysTime - ToA + {}^{DL}/_{2} & \text{for 2-hop node} \end{cases} \tag{2}$$

where *sysTime* is the local time at which the corresponding node finishes receiving the SYN message.

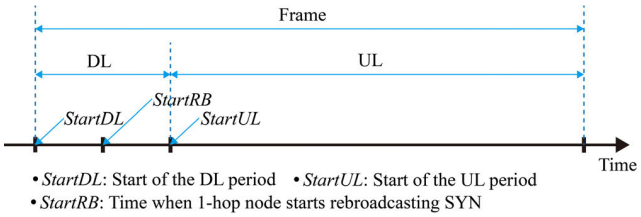


FIGURE 6. Time synchronization.

D. SLOT SCHEDULING

We assume that every node has one task. Given gateway g that covers n 1-hop nodes, the GW maintains *network profile information* $NPI(g)$ as follows:

$$NPI(g) = (pf_1, pf_2, \ldots, pf_n)$$

where pf_i as the *profile information* of 1-hop node i, expressed as follows:

$$pf_i = (\tau_i, \tau_{i1}, \ldots, \tau_{ij}, \ldots, \tau_{iC_i})$$

where τ_{ij} indicates the task of the j^{th} child of node i, and c_i is the number of children for node i.

Two-hop tree slot scheduling can be performed based on $NPI(g)$ using the frame-slot architecture and the LSI algorithm. Basically, an 1-hop node needs one slot, whereas a 2-hop node needs two slots to send one packet. Let us use two notations, $SD^1(i)$ and $SD^2(i)$, to indicate the slot demands of 1-hop node i and 2-hop node i, respectively. Then, the *total slot demand*, $TSD(i)$, of 1-hop node i is given as the sum of the slot demands needed by node i and all its children:

$$TSD(i) = SD^1(i) + \sum\nolimits_{j \in C(i)} SD^2(j) \tag{3}$$

where $SD^1(i) = 2^{class(i)}$ and $SD^2(j) = 2{*}2^{class(j)}$.

A node can obtain the demanded slots by using the logical slot indices if it knows its start logical slot index. Node i can calculate its *start logical slot index*, $startLSI(i)$, as follows:

$$startLSI(i) = \sum\nolimits_{j=1}^{i-1} TSD(j) + 1 \tag{4}$$

Naturally, the start logical slot index of node i is obtained by increasing *unity* assuming that all preceding nodes have obtained their demanded slots in $NPI(g)$. Then, node i generates a *slot allocation list*, $Alloc(i)$, for itself and $Alloc(j)$ for each node j in $C(i)$ according to Algorithm 2.

Algorithm 2 Get Slot Allocation Lists

// i: is the i^{th} node in $NPI(g)$
// lsi: a logical slot index
1 calculate $startLSI(i)$ according to Eq. (4);
2 $Alloc(i)$ = a sorted list of physical slot indices that correspond to the $SD^1(i)$ logical slot indices starting with $startLSI(i)$;
3 $lsi = startLSI(i) + SD^1(i)$;
4 **for** each $j \in C(i)$ **do**
5 $Alloc(j)$ = a sorted list of physical slot indices that correspond to the $SD^2(j)$ logical slot indices starting with lsi;
6 $lsi = lsi + SD^2(j)$;
7 **endFor**

Based on the slot allocation lists, 1-hop relay node i can generate a *local slot schedule*, $LSS(i)$, for itself and its children in $C(i)$ as follows:

$$\text{For relay node}i,\ LSS(i) = (TxSlots(i), RxSlots(i))$$

where $TxSlots(i)$ and $RxSlots(i)$ can be obtained as follows:

$$\begin{aligned} &TxSlots(i) = ascsort(Alloc(i) \cup \{x | x \in Alloc(j) \\ &\text{s.t. } (pos(x)\%2) = 0, \quad j \in C(i)\});\ and \\ &RxSlots(i) = \{x | x \in Alloc(j) \\ &\text{s.t. } (pos(x)\%2) = 1, \quad j \in C(i)\} \end{aligned} \tag{5}$$

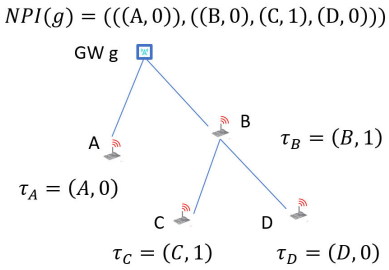
where $ascsort(L)$ indicates the *ascending-sorted list* of list L, and $pos(x)$ indicates the position of slot x in the corresponding list or set.

$$\text{For each } j \in C(i), \quad LSS(j) = (TxSlots(j))$$

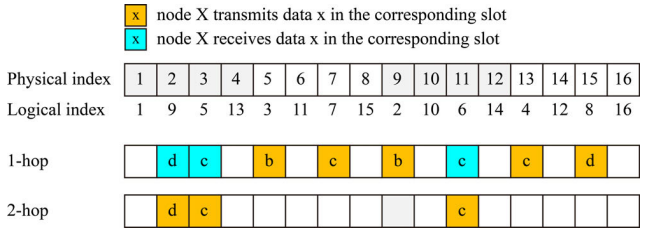
where $TxSlots(j)$ can be obtained as follows:

$$TxSlots(j) = \{x | x \in Alloc(j) \text{ s.t. } (pos(x)\%2) = 1\} \tag{6}$$

For relay node i, $TxSlots(i)$ is the ascending sorted list of all the physical slots that node i uses to transmit its own data and



(a) An example of a two-hop LoRa network



(b) Task scheduling for the network in (a)

FIGURE 7. An example of task scheduling in a two-hop LoRa network.

the data delivered from each child in $C(i)$. $RxSlots(i)$ includes all the physical slots that node i uses to receive data from its children, given as $\in RxSlots(i) = \bigcup_{jC(i)} TxSlots(j)$. For node j in C(i), $TxSlots(j)$ includes the physical slots that node j uses to transmit its own data.

Let us examine the scheduling for node B in Fig. 7(a), where $NPI(g) = (((A, 0)), ((B, 0), (C, 1), (D, 0)))$. Since C(A) = {}, C(B) = {C, D}, TSD(A) = 1, TSD(B) = 8, and $startLSI$(B) = 2, $Alloc$(B) = (5, 9), $Alloc$(C) = (3, 7, 11, 13), and $Alloc$(D) = (2, 15) according to Algorithm 2. Since $TxSlots$(B) = (5, 7, 9, 13, 15), $RxSlots$(B) = (2, 3, 11), $TxSlots$(C) = (3, 11), and $TxSlots$(D) = (2), LSS(B) = ((5, 7, 9, 13, 15), (2, 3, 11)), LSS(C) = ((3, 11)), and LSS(D) = ((2)). The corresponding slot schedule is illustrated in Fig. 7(b).

E. TWO-HOP DATA TRANSMISSION

1) DATA AGGREGATION

If every node transmits data according to the scheduled transmission slots, a relay node can consume much more energy than 2-hop nodes, since it has to receive all the data from its children and forward them to the GW. According to the datasheet of the LoRa SX1276 [25], shown in Table 2, a node consumes 28 mA in sending mode, much higher than the 10.3 mA in receiving mode.

In this section, an *optimal slot selection* (OSS) algorithm is designed to select the minimal number of transmission slots, thereby maximizing data aggregation at a relay node. The design of the OSS algorithm is based on the basic principle that every node delays its transmission as late as possible under the condition that it still should be able to transmit its packet in any of the scheduled slots to complete its transmission no later than the end of its transmission period. For this,

TABLE 2. Energy consumption subject to the modes of a LoRa SX1276 transceiver.

Node mode	Supply current in LoRa SX1276
Sending	28 (mA) at $P_{tx} = 13dBm$
Receiving	10.3 (mA) at BW = 125 kHz
Idle	1.5 (μA)
Sleep	0.2 (μA)

every node needs to know the latest slot among the allocated slots within its transmission period.

Given the slot allocation lists, every node can select the physical slots in which it must transmit based on its data transmission periods. For the design of the OSS algorithm, we define the *deadline list* (DL) for a relay node.

Suppose that node s has the shortest transmission period among relay r and its children in $C(r)$. Let P_x denote the period of node x. Then, deadline list $DL(r)$ for relay node r can be obtained as follows:

$$DL(r) = \{p | p = a \times P_s, p \leq UL, a = 1 \ldots k, k \text{ is an integer}\} \quad (7)$$

where UL is the uplink period in the slots. $DL(r)$ indicates the list of the last physical slot numbers in all periods of task s within one frame.

For example, both tasks τ_B= (B, 1) and τ_C = (C, 1) have the shortest period of 8 in Fig. 7(a). Then, for relay node B, DL(B) = (8, 16). For relay node r and $DL(r)$, the OSS algorithm is detailed in Algorithm 3.

Algorithm 3 Optimal Slot Selection

// $x.TxFlag$: if this is *True*, a relay must transmit data in slot x
// r: a relay node
1 **for** each $x \in TxSlots(r)$ **do**
2 $x.TxFlag = False$;
3 **endFor**
4 **for** each $k \in DL(r)$ **do**
6 $x = \max\{y | y \in TxSlots(r), y <= k\}$
6 $x.TxFlag = True$;
7 **endFor**

This algorithm is quite simple, since relay node r selects the latest transmission slot before missing each deadline in $DL(r)$. For node B in Fig. 7(a), $TxSlots$(B) = (5, 7, 9, 13, 15), where the underlined physical slot indices are the slots in which relay B must transmit aggregated data. However, the aggregated data may exceed the media access control (MAC) protocol data unit (MPDU) if a relay node waits until it meets the selected transmission slot. Thus, a relay node has to judge whether it has to transmit partially aggregated data or not *on-the-fly* by checking the size of the aggregated data. If the size of the currently aggregated data exceeds MPDU, it must transmit the aggregated data in the current slot to start queuing again.

Algorithm 4 Data Transmission

At 1-Hop (Relay) Node r	At 2-Hop Node x
1 Upon receiving *SYN* from GW: // initialization	1 Upon receiving *SYN* from a relay: //initialization
2 set $startULTimer = StartUL - sysTime$;	2 set $startULTimer = StartUL - sysTime$;
3 When *startDLtimer* expires:	3 When *startDLTimer* expires:
4 $StartUL = StartUL + UL + DL$;	4 $StartUL = StartUL + UL + DL$;
5 set $startULTimer = StartUL - sysTime$;	5 set $startULTimer = StartUL - sysTime$;
6 wait for command;	6 wait for command;
7 Upon receiving a *command* or *SYN* from GW:	7 Upon receiving a command or *SYN* from a relay:
8 process command;	8 process command;
9 sleep;	9 sleep;
10 When *startULTimer* expires:	10 When *startULTimer* expires:
11 $txseq = 1$; $rxseq = 1$;	11 $txseq = 1$;
12 $txsn = TxSlots(r)[txseq]$; $rxsn = RxSlots(r)[rxseq]$;	12 $txsn = TxSlots(x)[txseq]$;
13 set $TxTimer = (txsn - 1) * slotLen$;	13 set $TxTimer = (txsn - 1) * slotLen$;
14 set $RxTimer = (rxsn - 1) * slotLen$;	14 sleep;
15 sleep;	15 When *TxTimer* expires:
16 When *TxTimer* expires:	16 send *myPkt*;
17 **if** $txsn \in Alloc(r)$ **then** enqueue *myPkt*;	17 **if** $txseq <= |TxSlots(x)|$ **then**
18 aggregate all queued packets into *AggPkt*;	18 $txseq = txseq + 1$;
19 **if** ($txsn.TxFlag = True$) or	19 $txsn = TxSlots(x)[txseq]$;
20 $(MPDU - size(AggPkt) < size(Pkt))$ **then**	20 set $TxTimer = StartUL + (txsn - 1) * slotLen - sysTime$;
21 send *AggPkt*; clear queue;	21 **else**
22 **endIf**	22 $StartDL = StartUL + UL$;
23 **if** $txseq <= |TxSlots(r)|$ **then**	23 set $startDLTimer = StartDL - sysTime$;
24 $txseq = txseq + 1$; $txsn = TxSlots(r)[txseq]$;	24 **endIf**
25 set $TxTimer = StartUL + (txsn - 1) * slotLen - sysTime$;	25 sleep;
26 **else**	
27 $StartDL = StartUL + UL$;	
28 set $startDLTimer = StartDL - sysTime$;	
29 **endIf**	
30 sleep;	
31 When *RxTimer* expires:	
32 wait for a packet;	
33 **if** r receives *Pkt* **then** enqueue *Pkt*;	
34 **if** $rxseq <= |RxSlot(r)|$ **then**	
35 $rxseq = rxseq + 1$; $rxsn = TxSlots(r)[rxseq]$;	
36 set $RxTimer = StartUL + (rxsn - 1) * slotLen - sysTime$;	
37 **endIf**;	
38 sleep;	

MPDU	MAC protocol data unit
Pkt	A packet
myPkt	A packet the node generates itself
AggPkt	An aggregated packet
rxseq	Used as an index for the *RxSlots* list
txseq	Used as an index for the *TxSlots* list
rxsn	A variable for *Rx* slot number
txsn	A variable for *Tx* slot number
$X[i]$	The i^{th} element of array or list X
slotLen	The time length of a slot
sysTime	System time at the processing point

In addition, a filtering technique can be employed. A relay may discard some data by analyzing its data or the received data before aggregation. This can further reduce the number of transmissions, or can reduce the size of aggregated data.

2) DATA TRANSMISSION

Based on $TxSlots(i)$ and $RxSlots(i)$ for node i, given in Eqs. (5) and (6), Algorithm 4 explains how 1-hop (relay) nodes and 2-hop nodes perform data transmission with data aggregation.

Upon receiving SYN from the GW or a relay, every node performs time synchronization, calculates *StartUL*, and goes to sleep. Then, every node wakes up at *StartUL* to start data transmission according to the slot schedule. For 1-hop nodes, as soon as a node enters the UL period, it calculates the next transmission time from the first slot in $TxSlots(r)$ in line 13, and calculates the next receiving time from the first slot in $RxSlots(r)$ in line 14. Then, every node wakes at the next scheduled slot to transmit (line 16) or receive a packet (line 31). A relay node performs data aggregation until it must

TABLE 3. Comparison of the features of multi-hop LoRa protocols.

Features	LoRaBlink [19]	CT-LoRa [20]	LoRa-Mesh [21]	Sync-LoRa-Mesh [22]	Two-hop RT-LoRa
Reliability	Data acquisition using directed flooding that exploits the notion of concurrent non-destructive transmission	A flooding-based data acquisition that exploits the notion of timing offset-based concurrent transmission	Data acquisition by constructing a reliable tree and issuing a query to individual nodes	Data acquisition using slot scheduling for underground nodes, but using LoRaWAN for forwarding data to a gateway	Data acquisition using slot scheduling that takes into account signal attenuation by distance and obstruction
	High	High	High	High only for underground nodes	High
Real-time support	Not mentioned	Real-time scheduler can be used easily	Not mentioned	Not supported	Uses a real-time schedule for all nodes
	Not mentioned	Yes	No	No	Yes
Supported number of wireless hops	No limitation	No limitation	No limitation	No limitation	Two hops, but can support up to five hops with two gateways
Energy consumption	Every node has a low duty cycle, but remains active for one flooding-based data transmission	Every node has a low duty cycle, but remains active during each scheduled slot for one flooding-based data transmission	All nodes remain active during query to set up a path, and the nodes on the path remain active during data transmission	All subnet nodes remain active during the scheduled receiving slots and transmitting slots	Every node remains active only during the scheduled slots
	Mid, but high in high traffic	Mid, but high in high traffic	High	Mid	Low and optimized
Overhead	Transmission of beacon and data transmission by flooding	Transmission of slot schedule and data by flooding	Query based data acquisition	Subnet slot scheduling for synchronous data acquisition	One-time distribution of minimal task profile information
	High	High	High	Mid	Low
No. of supporting nodes	Not mentioned, but suitable for low traffic	Not mentioned, but suitable for low traffic	Not mentioned, but suitable for low traffic	Limited of number of nodes in a subnet subject to the maximum packet size of LoRaWAN	Maximal support subject to one transmission slot length

transmit a packet (line 19) by checking *TxFlag* or the size of an aggregated packet that may reach the maximum possible size (line 20). Furthermore, every 1-hop node always waits for a packet from its children (line 31). For 2-hop nodes, as soon as a node enters the UL period, it calculates the next transmission time from the first slot in $TxSlots(x)$ (line 13). Then, whenever a node finishes sending a packet, if it has more slots, it calculates the next transmission slot (line 19); otherwise, it goes into the next DL period (line 23).

To prevent external interference, the proposed protocol employs CAD [26] to implement the LBT mechanism. Thus, a node always senses the channel before its data transmission, and sets a random delay if the channel is busy. In general, every node may assume that its signal will be stronger than any external signal. Thus, even with the existence of external interference, a node can receive a packet with a high probability of success by exploiting the capture effect [27].

The GW evaluates the condition of the network periodically by analyzing the number of successfully received data packets from the participating nodes. If it fails to receive data from a certain portion of the nodes, it can initiate network maintenance by broadcasting a *request network maintenance* command during the DL period. Then, at the end of the next UL period, every node starts the NM period.

IV. PROTOCOL EVALUATION

A. COMPARISION OF CHARACTERISTICS FOR MULTI-HOP LoRa PROTOCOLS

Table 3 compares the features of different multi-hop LoRa protocols. Two-hop RT-LoRa allocates distinct transmission slots to each node by considering signal attenuation of participating nodes, and it allows every node to transmit data within the allocated transmission slots, thereby enabling reliable data transmission. LoRaBlink allows slotted channel access by flooding the network with a beacon message for time synchronization, and it also enables a node to transmit data only at the boundary of the slots. It exploits the notion of concurrent transmission, since multiple nodes can broadcast a beacon message or data simultaneously. However, this approach can still experience collisions if a receiver receives multiple data from two or more nodes simultaneously, where the difference in receiving power is less than a certain threshold. CT-LoRa tries to improve the reliability of data transmissions by giving different time offsets to different concurrent transmitters. However, these two approaches cause high overhead, since they involve the flooding of messages and data. Thus, this may not be suitable for industrial monitoring and control applications with relatively heavy data traffic. LoRa-Mesh issues a query message whenever a gateway wants to get data

from a particular node. In sensor networks, every node tends to send one datum to a gateway periodically. Thus, this way of data acquisition can cause the transmission of too many control messages for data acquisition. In Sync-LoRa-Mesh, a relay node establishes a reliable tree with underground nodes and acquires data from those nodes synchronously by scheduling slots, and then acts as one of the LoRaWAN nodes to forward the collected data to a gateway.

In Two-hop RT-LoRa, nodes share their data transmission periods with other nodes via a server so that each node can be allocated transmission slots such that it meets its time constraint. In CT-LoRa, a gateway can generate a real-time schedule for the nodes, and every node can send data by means of flooding without interference within the allocated slot. Other approaches do not consider real-time data transmission. These protocols (except Two-hop RT-LoRa) do not limit the number of supportable wireless hops and nodes, whereas Sync-LoRa-Mesh limits the number of nodes that a relay can support under a constraint on the data size that LoRaWAN allows in a single transmission. In Two-hop RT-LoRa, one gateway allows only two wireless hops, since more wireless hops can lead to higher amounts of data traffic. Note that data in a 2-hop node has to be transmitted twice to reach the gateway, and each transmission can easily interfere with other transmissions in long-range LoRa networks. In CT-LoRa, data are transmitted to the gateway using flooding. All nodes have to remain active during data transmission, and they retransmit data in a concurrent manner. In LoRaBlink, nodes control data retransmissions using the hop distance to the gateway such that a node retransmits data only if it has a shorter hop distance to the gateway than the sender. Two-hop RT-LoRa optimizes energy consumption, since it allows a node to remain active during the assigned slots, and it uses the optimal aggregated transmission. One most important feature of Two-hop RT-LoRa is using slot scheduling-based data transmission that does not allow data collisions. Therefore, if a slot size is optimized, the optimal number of nodes can be supported. In LoRaBlink and CT-LoRa, flooding time becomes a duty cycle that is optimized by exploiting concurrent transmission. Nevertheless, a single datum is retransmitted many times by many nodes until it arrives at the gateway.

According to the discussion so far, Two-hop RT-LoRa enables energy-efficient, reliable, real-time transmission, and can support a considerable number of nodes by completely removing the possibility of collision. In consequence, it can satisfy the requirements of industrial applications.

B. ANALYSIS

1) NUMBER OF SUPPORTABLE NODES

The UL data format consists of a preamble, the payload, and a 16-bit payload cyclic redundancy check (CRC). Suppose that $symbols(X)$ and $bytes(X)$ denote the number of symbols and bytes of message X, respectively. Then, the preamble duration, $T_{preamble}$, and the payload duration, $T_{payload}$, are given as follows:

$$\begin{aligned} T_{preamble} &= (symbols(preamble) + 4.25)^* T_{sym}, \text{ and} \\ T_{payload} &= (symbols(payload + CRC))^* T_{sym} \end{aligned} \tag{8}$$

where, $T_{sym} = {2^{SF}}/{BW}$, and 4.25 indicates a fixed number of symbols including two up-chirps, two down-chirps, and a 4 up-chirp that are transmitted in the preamble on-air according to the LoRa modulation specifications, and $symbols(payload + CRC)$ is given as follows [25].

$$\begin{aligned} &symbols(payload + CRC) \\ &= 8 + \max(ceil\left[\frac{8PL - 4SF + 28 + 16 - 20H}{4(SF - 2DE)}\right] \\ &\quad \times (CR + 4), 0) \end{aligned} \tag{9}$$

where, $PL = bytes\ (payload)$, SF is a spreading factor, $H = 0$ and $H = 1$ when a header is enabled and disabled, respectively, $DE = 0$ and $DE = 1$ when low data–rate optimization is enabled and disabled, respectively, and CR is the coding rate in [1, 4].

The lower bound of the UL slot length, $slotLen_{LB}$, is given as follows:

$$slotLen_{LB} \geq T_{preamble} + T_{payload} \tag{10}$$

Then, the lower bounds of the UL slot length for different payloads with the parameters $symbols(preamble) = 8$, $SF = 7$, $BW = 125$, *and* $CR = 1$ are summarized in Table 4, where $T_{slot} \geq 107.78\ ms$ when $PL = 60$.

TABLE 4. The lower bound of the UL slot based on payload (SF = 7, BW = 125).

PL (bytes)	30	60	90	120
$slotLen_{LB}(ms)$	66.82	107.78	153.86	199.94

Let α be the proportion of 1-hop nodes in the network ($0 < \alpha \leq 1$). For simplicity, assume that every node generates one packet during a frame period. Then, the network slot demand, $NetSD(n)$, that n nodes require during a frame period is given as follows:

$$NetSD(n) = \alpha^* n + 2(1 - \alpha)^* n = (2 - \alpha)^* n \leq 2^N \tag{11}$$

In other words, *nSupportableNodes* is the number of nodes that the network can support on a single channel, given as follows:

$$nSupportableNodes = n \leq {2^N}/{(2 - \alpha)} \tag{12}$$

Based on Eq. (12), Fig. 8 illustrates the number of nodes that a single-channel GW can support with varying N and α when PL = 30 bytes and SF = 7. It shows that *nSupportableNodes* increases linearly with α. If N = 8 and α = 0.7, a single-channel GW can support 200 nodes. Note that the minimum frame size with N = 8 (= 256 slots) will become 17,105 ms. In Fig. 9, *nSupportableNodes* decreases logarithmically according to the increase in SF with N = 8 and α = 0.7. This is because the increase in SF by *unity* doubles the slot length.

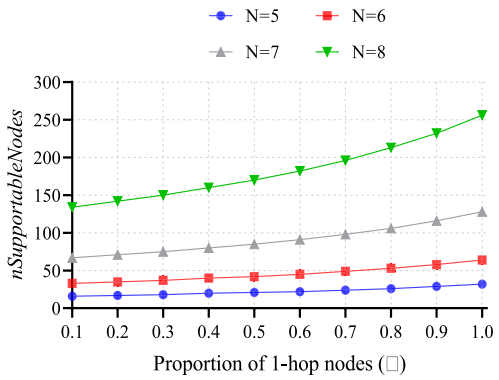


FIGURE 8. Number of supportable nodes with varying α value (BW=125, CR=1, SF=7).

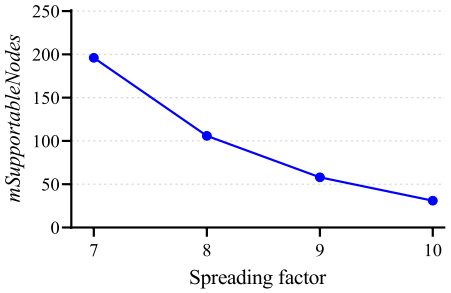


FIGURE 9. Number of supportable nodes for different SFs (BW=125, CR=1, N = 8, α = 0.7).

2) ENERGY CONSUMPTION

In our approach, data can be transmitted in two hops, rather than one hop, to a gateway to improve reliability. We analyze how this way of data transmission can affect energy consumption by comparing energy consumption in a two-hop transmission and in a one-hop transmission. E_{twohop} and E_{onehop}, as energy consumptions for two-hop transmission and one-hop transmission, respectively, can be expressed as follows:

$$E_{twohop} = (2^{*}P_{Tx} + P_{Rx})^{*}PktToA(SFi), \text{ and} \quad (13)$$

$$E_{onehop} = P^{*}_{Tx}PktToA(SFi) \quad (14)$$

where P_{Tx} and P_{Rx} indicate sending power and receiving power, respectively, and $PktToA(SFi)$ indicates the time on air of a transmitted packet when SFi is used.

In this analysis, the Semtech SX1276 working at 868 MHz is used as an energy reference model [25] where, at 25°C with input voltage = 3.3V, the supply current values for a transmitter vary according to transmission power, and those for the receiver vary according to the bandwidth, as seen in Table 5.

TABLE 5. Supply current values for a transmitter.

Transmitter	$P_{Tx}(dBm)$	7	13	17	
	$I_{Tx}(mA)$	20	28	90	
Receiver	BW (kHz)	≤62.5	125	250	500
	$I_{Rx}(mA)$	9.9	10.3	11.1	12.6

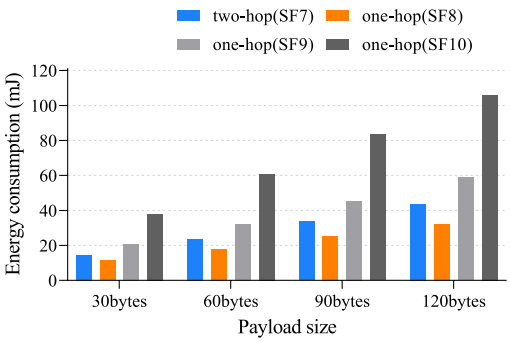


FIGURE 10. Comparison of energy consumption for one-hop and two-hop transmissions with varying payload sizes ($BW = 125kHz$,[[space]]$CR = 1$ and $P_{tx} = 13dBm$).

Based on Eqs. (13) and (14), Fig. 10 compares energy consumption for two-hop(SF7), which transmits a packet in two hops with SF7, and for one-hop(SF8), one-hop(SF9), and one-hop(SF10), which transmit packets directly to the gateway using SF8, SF9, and SF10, respectively, in terms of energy consumption when varying the size of the payload. The energy consumption of two-hop(SF7) is higher than that of one-hop(SF8) to some degree, but the difference is not significant; however, two-hop(SF7) consumes far less energy than both one-hop(SF9) and one-hop(SF10). Thus, we conclude that it is desirable to use two-hop(SF7) for industrial zones in which nodes can experience high signal attenuation.

C. EVALUATION

1) EVALUATION OF DATA AGGREGATION

The proposed data aggregation approach is evaluated with a small network that consists of one GW, one 1-hop node, and three 2-hop nodes, using frame size N = 5 (= 32 slots). Each 1-hop node randomly selects its task class, from 0 to 3, under the constraint that the total slot demand is smaller than or equal to the frame size.

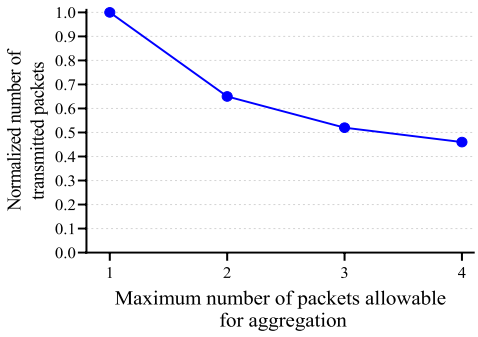


FIGURE 11. The normalized number of transmitted packets subject to the maximum number of aggregated packets limited by the MPDU.

Fig. 11 illustrates the ratio of the number of packets transmitted by an 1-hop node to the total packets generated by 1-hop and 2-hop nodes. The number of transmissions at an 1-hop node decreases rapidly as the number of packets allowable for aggregation increases. It shows that if four packets are

allowed for aggregation, the number of transmitted packets can be reduced by over 50%.

2) EVALUATION OF RELIABILITY

a: EXPERIMENT SETUP

A testbed that consisted of one GW and 10 end nodes was constructed for an experiment on the campus of the University of Ulsan, where the GW was placed inside the laboratory of the computer engineering building, and the end nodes were positioned manually, such that some radio signals were highly attenuated by multiple layers of concrete walls and buildings. Every node transmitted a packet of 30 bytes per data-collection cycle of six seconds. The experiment was performed over 500 data-collection cycles. The key parameters and values are summarized in Table 6.

TABLE 6. Experiment parameters.

Parameter	Value	Parameter	Value
Number of nodes	10	Spreading factor	SF7
Payload size	30 bytes	Bandwidth	125 kHz
Number of channels	1	Code rate	4/5
Tx power	14 dBm	Preamble	8 symbols

b: EXPERIMENT RESULTS

Fig. 12 shows a map of the end nodes deployed on campus after network construction, where the GW was located inside our laboratory, nodes 1, 2, 3, 8, 9, and 10 were inside buildings, and nodes 4, 5, 6, and 7 were in open areas of the campus. The solid lines and the dashed lines indicate stable, unstable (or bad) links to the GW, respectively. The nodes with bad link quality became 2-hop nodes that connected to an 1-hop relay node. For example, nodes 7 and 8 became 2-hop nodes that used node 5 as an 1-hop relay.

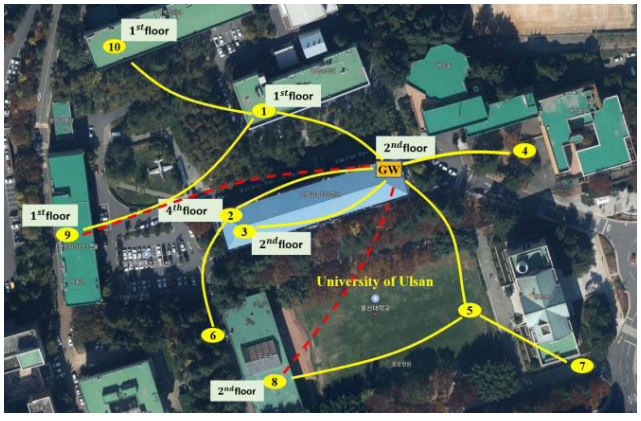


FIGURE 12. The testbed deployed on the University of Ulsan campus.

Experiments were performed by assigning SF7 to every node, and the results are shown in Table 7. With construction of a two-hop network, the gateway could achieve PRRs above 95% for 1-hop nodes, and above 94% for 2-hop nodes. The PRRs for 2-hop nodes are clearly compared with those for the same nodes that transmitted packets directly to the GW (given in parentheses).

TABLE 7. Experimental results from using SF7.

Node ID	Hop distance	PRR (%)
1	1	99
2	1	99
3	1	95
4	1	100
5	1	98
6	2	95 (0)
7	2	94 (0)
8	2	97 (28)
9	2	98 (11)
10	2	97 (0)

Note: The value in parenthesis indicates PRR when the node transmitted a packet directly to the GW

More experiments were performed by having the 2-hop nodes transmit packets directly (i.e., in a one-hop transmission) to the GW by increasing the SF, as shown in Fig. 13. In that figure, two-hop(SF7) and one-hop(SFx) indicate the PRRs when a two-hop transmission was executed with SF7 and when a one-hop direct transmission was executed by using SFx. Even though the 2-hop nodes used the higher SF, say SF10, the improvement in PRR was limited, showing that nodes 6 and 7 still failed to send packets. This implies that two-hop transmission is much more effective in terms of energy consumption and transmission reliability.

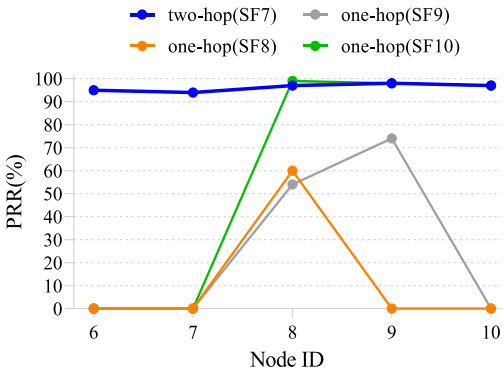


FIGURE 13. PRRs with 2-hop nodes using different SFs.

V. CONCLUSION

A two-hop RT-LoRa protocol was proposed for use in industrial monitoring and control systems that require real-time, reliable, and energy-efficient data transmission. The crux of the approach lies in the construction of a reliable two-hop tree, distributed slot scheduling of low complexity, and optimal data aggregation. In real-time IoT applications, the proposed protocol can allocate slots easily to 1-hop and 2-hop end nodes such that every node meets its time constraint

if it transmits packets in the allocated slots. Furthermore, the protocol can overcome the problem of packet loss from signal attenuation by enabling two-hop transmission. For energy saving, an 1-hop relay node can minimize the number of transmissions by using data aggregation. It was shown by analysis that our protocol can support hundreds of nodes on a single channel. Since the protocol uses a slot-scheduled approach, it can achieve high reliability in data transmissions, regardless of the number of deployed nodes.

REFERENCES

[1] *LoRaWAN TM 1.1 Specification*, L. Alliance, Fremont, CA, USA, 2017.

[2] *A Technical Overview of LoRa and LoRaWAN*, L. Alliance, Fremont, CA, USA, 2015.

[3] A. Augustin, J. Yi, T. H. Clausen, and W. Townsley, "A study of LoRa: Long range & low power networks for the Internet of Things," *Sensors*, vol. 16, p. 1466, Sep. 2016.

[4] N. Abramson, "THE ALOHA SYSTEM: Another alternative for computer communications," in *Proc. Fall Joint Comput. Conf.*, vol. 37, Nov. 1970, pp. 281–285.

[5] M. Bor, U. Roedig, T. Voigt, and J. Alonso, "Do LoRa low-power wide-area networks scale?" in *Proc. 19th ACM Int. Conf. Modeling, Anal. Simulation Wireless Mobile Syst.*, 2016, pp. 59–67.

[6] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, "Understanding the limits of LoRaWAN," *IEEE Commun. Mag.*, vol. 55, no. 9, pp. 34–40, 2017.

[7] O. Georgiou and U. Raza, "Low power wide area network analysis: Can LoRa scale?" *IEEE Wireless Commun. Lett.*, vol. 6, no. 2, pp. 162–165, Apr. 2017.

[8] F. Van den Abeele, J. Haxhibeqiri, I. Moerman, and J. Hoebeke, "Scalability analysis of large-scale LoRaWAN networks in ns-3," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 2186–2198, Dec. 2017.

[9] A. M. Yousuf, E. M. Rochester, B. Ousat, and M. Ghaderi, "Throughput, coverage and scalability of LoRa LPWAN for Internet of Things," in *Proc. IEEE/ACM 26th Int. Symp. Qual. Service (IWQoS)*, Jun. 2018, pp. 1–10.

[10] B. Reynders, Q. Wang, P. Tuset-Peiro, X. Vilajosana, and S. Pollin, "Improving reliability and scalability of LoRaWANs through lightweight scheduling," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1830–1842, Jun. 2018.

[11] M. Rizzi, P. Ferrari, A. Flammini, E. Sisinni, and M. Gidlund, "Using LoRa for industrial wireless networks," in *Proc. IEEE 13th Int. Workshop Factory Commun. Syst. (WFCS)*, May 2017, pp. 1–4.

[12] T. Polonelli, D. Brunelli, A. Marzocchi, and L. Benini, "Slotted ALOHA on LoRaWAN-design, analysis, and deployment," *Sensors*, vol. 19, no. 4, p. 838, Feb. 2019.

[13] R. Piyare, A. Murphy, M. Magno, and L. Benini, "On-demand LoRa: Asynchronous TDMA for energy efficient and low latency communication in IoT," *Sensors*, vol. 18, no. 11, p. 3718, Nov. 2018.

[14] *Symphony Link TM Protocol*, Link Labs, Annapolis, MD, USA, 2015.

[15] J. Haxhibeqiri, I. Moerman, and J. Hoebeke, "Low overhead scheduling of LoRa transmissions for improved scalability," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3097–3109, Apr. 2019.

[16] D. De Guglielmo, S. Brienza, and G. Anastasi, "IEEE 802.15.4e: A survey," *Comput. Commun.*, vol. 88, pp. 1–24, Aug. 2016.

[17] M. Cattani, C. Boano, and K. Römer, "An experimental evaluation of the reliability of LoRa long-range low-power wireless communication," *J. Sens. Actuator Netw.*, vol. 6, no. 2, p. 7, Jun. 2017.

[18] J. Haxhibeqiri, A. Karaagac, F. Van den Abeele, W. Joseph, I. Moerman, and J. Hoebeke, "LoRa indoor coverage and performance in an industrial environment: Case study," in *Proc. 22nd IEEE Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2017, pp. 1–8.

[19] M. Bor, J. Vidler, and U. Roedig, "LoRa for the Internet of Things," in *Proc. EWSN*, Feb. 2016, pp. 361–366.

[20] C.-H. Liao, G. Zhu, D. Kuwabara, M. Suzuki, and H. Morikawa, "Multi-hop LoRa networks enabled by concurrent transmission," *IEEE Access*, vol. 5, pp. 21430–21446, 2017.

[21] H. C. Lee and K.-H. Ke, "Monitoring of large-area IoT sensors using a LoRa wireless mesh network system: Design and evaluation," *IEEE Trans. Instrum. Meas.*, vol. 67, no. 9, pp. 2177–2187, Mar. 2018.

[22] C. Ebi, F. Schaltegger, A. Rust, and F. Blumensaat, "Synchronous LoRa mesh network to monitor processes in underground infrastructure," *IEEE Access*, vol. 7, pp. 57663–57677, 2019.

[23] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient network flooding and time synchronization with Glossy," in *Proc. 10th ACM/IEEE Int. Conf. Inf. Process. Sensor Netw.*, Apr. 2011, pp. 73–84.

[24] Q. L. Hoang, W.-S. Jung, T. Yoon, D. Yoo, and H. Oh, "A real-time LoRa protocol for industrial monitoring and control systems," *IEEE Access*, vol. 8, pp. 44727–44738, 2020.

[25] *SX1276/77/78/79-137 MHz to 1020 MHz Low Power Long Range Transceiver—DATASHEET*, Semtech, Camarillo, CA, USA, 2019.

[26] *AN1200.21 Reading Channel RSSI During a CAD*, Semtech, Camarillo, CA, USA, 2014.

[27] U. Noreen, L. Clavier, and A. Bounceur, "LoRa-like CSS-based PHY layer, capture effect and serial interference cancellation," in *Proc. 24th Eur. Wireless Conf.*, May 2018, pp. 1–6.

HUU PHI TRAN received the B.S. degree from the School of Telecommunication Engineering, Xidian University, Xi'an, China, in 2012. He is currently pursuing the Ph.D. degree with the Department of Computer Engineering, University of Ulsan, South Korea. His current research interests include embedded systems, wireless sensor networks, and low power wide area networks.

WOO-SUNG JUNG (Member, IEEE) received the dual B.S. degree in electrical and computer engineering, information and computer engineering, and the M.S. and Ph.D. degrees from the School of Computer Engineering, Ajou University, South Korea, in 2007, 2009, and 2015, respectively. He is currently a Senior Researcher with the Electronics and Telecommunications Research Institute (ETRI), Daejeon, South Korea. His current research interests are in wireless networking, the Internet of Things, device-to-device communication, and embedded systems.

TAEHYUN YOON received the B.S., M.S., and Ph.D. degrees from the School of Electronics Engineering, Kyungpook National University, in 2005, 2007, and 2015, respectively. Since 2016, he has been a Researcher with the Intelligent Robotics Research Division, Electronics and Telecommunications Research Institute (ETRI), Daejeon, South Korea. His current research interests include applied mobile communication, ship-IT convergence, and LoRa networks.

DAE-SEUNG YOO received the B.S., M.S., and Ph.D. degrees from the Department of Computer Engineering and Information Technology, Ulsan University, in 1998, 2001, and 2011, respectively. Since 2009, he has been a Research Engineer with the Intelligent Robotics Research Division, Electronics and Telecommunications Research Institute (ETRI), Daejeon, South Korea. His current research interests include applied software engineering, mobile communication, ship-IT convergence, and ad-hoc networks.

HOON OH (Member, IEEE) received the B.S.E.E. degree from Sung Kyun Kwan University, Seoul, and the M.Sc. and Ph.D. degrees in computer science from Texas A&M University, College Station, TX, USA, in 1993 and 1995, respectively. From 1983 to 1989 and 1996 to 2000, he worked as a Software Engineer and a Software Architect with the Corporate Research Center, Samsung Electronics Company Ltd. He was involved in developing communication protocols for the data services of the CDMA and IMT2000 handset products. He has been serving as a Department Head of the IT Convergence Department, University of Ulsan, South Korea, since 2016. He is currently a Professor with the Department of Computer Engineering and Information Technology and the Director of the Vehicle IT Convergence Technology Research Center, University of Ulsan. He has published over 55 refereed journals for the last decade and made many technology transfers to the local IT companies through University-industry cooperation program. His research interests include embedded systems, mobile ad hoc networks, real-time computing, and context-aware computing. He is a member of IEICE, and ICASE, and a Lifetime Member of KICS and KISA. He received a Best Paper Award from the National Academy of Science, USA, in 1995.