IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 31, 2020, accepted February 18, 2020, date of publication March 2, 2020, date of current version March 13, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2977659

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

QUY LAM HOANG[1], WOO-SUNG JUNG[2], TAEHYUN YOON[2], DAESEUNG 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, Daejon 305-350, South Korea

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

This work was supported in part by the Electronics and Telecommunications Research Institute (ETRI), and in part by the Korean Government and Ulsan Metropolitan City through the (Development of smart context-awareness foundation technique for major industry acceleration) under Grant 19ZS1300 and through the (Development of smart HSE system and digital cockpit system based on ICT convergence for enhanced major industry) under Grant 19AS1100.

ABSTRACT LoRa technology draws attention for its use in industrial monitoring and control systems in which each end device or task is required to send data periodically to a (cloud) server. Despite its provision of a stable link, it suffers from data loss by signal suppression and interference. A real-time LoRa protocol is proposed that uses a slot scheduling to remove collision and device or node grouping based on signal attenuation to deal with signal suppression. Based on the definition of a frame-slot structure, a logical slot indexing algorithm is devised to tag a logical index to each slot. The logical indices enable the easy allocation of slots to nodes such that if each node sends data in the allocated slots, it can satisfy time constraint. To handle external interference caused by other networks, the protocol uses a multiple listen-before-talk (mLBT) mechanism that allows channel detection multiple times within one slot. Our protocol is compared analytically and experimentally with other ones to show its superior throughput and reliability against signal suppression and interference.

INDEX TERMS LoRa protocol, real-time, reliability, task scheduling, TDMA.

I. INTRODUCTION

The use of wireless sensor networks (*WSN*s) in industrial monitoring and control systems (hereafter abbreviated as *IMOCS*s) has some requirements such that data has to be delivered in a real-time and reliable manner [1]. Intensive studies on WSNs have been conducted to satisfy those requirements [2]–[5]. However, multi-hop WSNs have some difficulties in dealing with the changes in network topology by node mobility and link instability due to shading and multipath fading effects [1]. In particular, this phenomenon becomes severer in underground tunnels or enclosed spaces. In [6], [7], the authors claim that the LoRa technology is suitable for the use in industry zones due to its high interference immunity and low power data transmission.

A private LoRa network for industry use, referred to as an *industrial LoRa network*, draws some issues. First, IMOCS collects data regularly from each end device (or *node*), thereby incurring relatively high traffic loads. For the worse, since LoRa data has a lengthy *time-on-air* (*ToA*), the industrial LoRa network suffers from high *internal interference* between different data transmissions in the same network. Second, in IMOCS, each node can have its own data transmission period (that becomes time constraint) to prevent erroneous operations from the temporal inaccuracy of sensor data. Third, high traffic in IMOCS also makes data transmission more vulnerable to *external interference* caused by other ISM band networks [1]. The first two issues can be tackled by the use of the time division multiple access (TDMA); however, the third issue has to be treated in a best-effort manner since external interference is not controllable. One derived issue with the use of TDMA is that if a slot scheduling is made in a centralized manner as it is done in most WSN protocols [8]–[10], it can cause high overhead against topological changes. The overhead may be tolerable in a LoRa network with a star topology; it is still a burden due to its low bandwidth.

Recently, LoRaWAN [11], an open network standard on top of the LoRa physical layer, has been used in various applications such as smart cities, smart farming, and

The associate editor coordinating the review of this manuscript and approving it for publication was Jihwan P. Choi.

environmental monitoring, etc. [12]. LoRaWAN uses a star-of-star network topology, consisting of one or more gateways (GWs) that relay data between end nodes and a server. It defines three MAC classes, namely A, B, and C, in which Class A is widely used. It employs the *Pure Aloha* [13] approach for data transmission that allows an end node to transmit uplink data freely. Two downlink slots are opened after the uplink slot for a server to send data or command to end nodes. Due to the random nature of data transmission, a LoRaWAN node suffers from the high probability of data collision as traffic increases. The collision problem with the scalability issue is studied in papers [14]–[17]. Furthermore, it obvious that the collision problem becomes worse with the use of the higher spreading factors (SF) that have the longer transmission range and the lengthier packet ToA.

A number of solutions have been given to tackle the collision problem. Some of them employs the *Slotted Aloha* [13] approach that constrains the start of data transmission only to the boundaries of time slots [18]–[20]. By simulation, they show that this approach can alleviate the probability of collision to some extent, but does not eliminate collision. Thus, Slotted Aloha may not be suitable for the applications that require high reliability in data transmission. Meanwhile, some protocols employ a *slot scheduling* to remove data collision completely. In the slot scheduling approach, every node is allocated a distinct time slot such that if it transmits data within the allocated slot, no data collision occurs [21], [22]. In [23], a relay node connected directly to GW creates a subnet with a small number of nodes in the underground zone and then collects data from them by using a slot schedule. However, it still relies on LoRaWAN to forward the collected data to GW. The on-demand LoRa protocol [24] allows a server to collect data from nodes in a cluster that is managed by a clusterhead. By using a short-range wake-up radio combined with the LoRa module, the clusterhead awakes its members and schedules transmission slots for them. Although this approach improves reliability, it can limit its application scenario since its operation relies on the short-range wake-up radio. The authors in [25] improve the concurrent transmission technique used in the Glossy approach [26] by employing the notion of transmission time offset, referred to as the *offset-CT* approach. This approach allows two or more nodes to transmit data concurrently, but with different time offsets; however, it suffers from high energy consumption since it involves flooding in every data transmission.

Several efforts have been made to adapt the LoRa technology to industry applications by employing a slot scheduling. The authors in [27] examine the applicability of the time-slotted channel hopping (TSCH) mechanism [10] with a small testbed that consists of one GW and three nodes using different SFs. However, they allow all transmitting nodes to have the same signal strength; thus, the experimental results do not reveal the effect of the imperfect orthogonality [28] on the transmitted signals. The imperfect orthogonality of signals with different SFs, also called *signal suppression effect*, allows a receiver to receive only the strongest signal unless the signals have the difference of signal strengths under a certain threshold [28]. In [29], the authors introduce a TDMA-based MAC protocol in which nodes transmit data to GW either in the Contention Access Period (CAP) section or in the Contention-Free Period (CFP) section. While the protocol employs the Aloha approach during CAP, it uses a slot schedule during CFP. This approach used an offline slot schedule in simulation without relying on any specific mechanism for slot assignment. In addition, they do not address the problem of the signal suppression effect that often makes parallel data transmission using different SFs impractical. In [30], a server collects MAC addresses from nodes and based on the analysis of those MAC addresses, determines a proper frame size such that if every node selects a slot in the frame by using its own MAC address in a distributed manner, no nodes are assigned the same slot. This approach can reduce scheduling overhead; however, it may waste many slots since it has to increase the frame size sufficiently so that every node can choose a distinct slot. Thus, this approach can degrade bandwidth efficiency. In addition, it does not consider time constraints.

In this paper, a real-time LoRa protocol is proposed that employs an efficient slot scheduling method to deal with the issues in industrial LoRa networks. In our protocol, a server collects data from every end device or *task* per a *data acquisition cycle time* referred to as a *frame*. A frame is divided into a number of *slots*. A real-time task scheduling algorithm is developed to generate a task schedule such that if a task transmits data in each allocated slot, it can satisfy its data transmission period. For the ease of scheduling, the task scheduling algorithm employs a *logical slot indexing algorithm* that assigns a logical slot index to each slot of the frame. It generates a task schedule by having each task select the required number of slots sequentially, starting with any logical slot index. In addition, an efficient distributed scheduling mechanism is presented to reduce scheduling overhead. To deal with the signal suppression effect, the nodes are divided into different groups based on the degree of signal attenuation. Then, each group is assigned a distinct SF and distinct frequency channel so that the nodes in different groups do not interfere each other. In data transmission, if a task gives up data transmission immediately when it detects a channel is busy within the allocated slot, it is too costly. Thus, the protocol employs a *multiple listen-before-talk* (*mLBT*) mechanism that allows channel detection multiple times within a slot.

Analysis was given to show the least upper bounds of network throughput for different approaches, where our approach shows significantly higher throughput bound than LoRaWAN and Slotted Aloha. Experiment was also performed on a testbed with a single-channel gateway and fifteen nodes. It is believed that a single channel is sufficient to reveal the key operational characteristics of our protocol. In this experiment, packet delivery rates (PDRs) for different approaches were compared to show the superiority

of the proposed protocol. The analysis of PDR was also given to verify the soundness of experimental results. The proposed protocol could far outperform LoRaWAN and Slotted Aloha in throughput and reliability. It can avoid the signal suppression effect that significantly degrades the PDRs of the compared protocols. In addition, the effectiveness about the use of *mLBT* was examined while some interfering nodes were generating garbage data to incur external interference intentionally.

The paper is organized as follows. Section II gives research background, and Section III describes the proposed protocol formally. Analysis and experiment are given in Section IV, and followed by concluding remarks in Section V.

II. BACKGROUND

A. NETWORK MODEL

LoRa technology uses the chirp spread spectrum (CSS) technique that supports a low data rate, but allows the demodulation for a signal of an extremely low strength and thus enables long range communication. To control the trade-off between data rate and transmission range, LoRa allows different SFs ranging between 7 and 12. A higher SF allows the lower data rate, but the longer transmission range [12]. Different SFs allows LoRa nodes to overcome signal attenuation caused by distance and obstruction in industrial environment [31], thereby enabling a star topology.

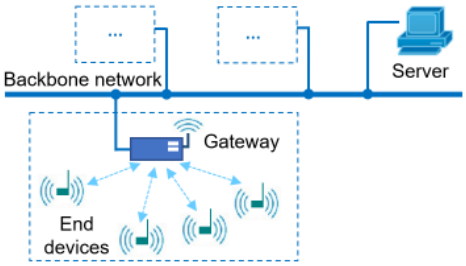


FIGURE 1. LoRa network model.

A LoRa network model considered for protocol design is assumed to be a star topology that has the direct connection between one gateway (GW) and each of end devices as illustrated in Fig. 1. A server is connected to the GW through a backbone network to collect and analyze the collected data. GW can receive multiple packets simultaneously from different channels [12]. Each node has one *task* as an active object for communication with GW. Each task is required to send one packet periodically via GW to the server. A server may send a control message to nodes via GW.

B. TERMINOLOGIES AND NOTATIONS

A *slot* is a scheduling unit whose size is sufficiently large to transmit one data packet. A *frame* is divided into a number of slots and is said to be *saturated* if all the slots in the frame are allocated to tasks. A group of one or more adjacent slots is called a *section*, and when the section is divided into two smaller parts of equal size, each divided part is also said to be a section. A *logical slot index* is assigned to each slot, apart from a physical slot index that exists inherently. When a section, say S ($= 2^N$ slots), is equally divided into 2^i sections, each of the divided sections is denoted by $S^i(\alpha)$, where α is the maximum logical index in the section. A section is said to be *index-free* if none of slots in this section is indexed. The maximum logical index of an index-free section is zero. Task τ_i is represented by its transmission period P_i in slots as follows:

$$\tau_i = (P_i) \tag{1}$$

This indicates that τ_i has to send one packet per period P_i.

C. MOTIVATION

In IMOCS, each task is required to send one data packet to a server per its own data transmission period. Since a task can have a transmission period shorter than the length of a frame, it can transmit data multiple times within one frame period. Suppose that a frame is divided into 2^N data transmission slots. If a task has its transmission period that corresponds to $2^N/2^k (0 \le k \le N)$ slots, it should be allocated 2^k slots within the frame such that each allocated slot appears only once per its transmission period. However, since different tasks can have different periods, it may not be easy to make the whole task set schedulable. If a brute-force scheduling method is employed, a slot schedule can be biased such that some tasks take all the early slots, thereby preventing other tasks from being scheduled.

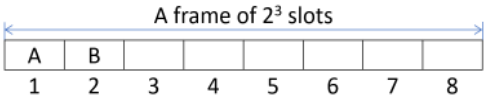


FIGURE 2. An example of a biased scheduling.

For example, consider a frame of 2^3 slots (N = 3) as depicted in Fig. 2 and three tasks A, B, and C that have the periods of 2^3, 2^3 and 2^1 slots, respectively. If tasks A and B take slots 1 and 2, respectively, task C that requires 4 slots cannot be feasibly scheduled since all the slots in the first quarter frame are not available.

The slot scheduling based approach includes some additional functions such as time synchronization and the generation, distribution, and maintenance of a slot schedule. With a star topology, time synchronization is relatively easy [32]; however, the implementation of other functions can be very costly in low bandwidth LoRa networks. One way to alleviate scheduling overhead is to divide nodes into groups so that a slot scheduling can be done for each group independently. Furthermore, if all nodes can know a *frame structure* and the periods of all tasks in the network, every node can generate the same slot schedule in distributed manner. The schedule distribution problem can be reduced to the distribution problem of task scheduling information. This can be done by sharing the ID and period of every task with other nodes. Furthermore, if each node determines its group based on the signal attenuation of a message received from GW and selects its SF based on its group, the problem of data loss by

signal attenuation can be alleviated considerably, considering the characteristics of the LoRa technology. The grouping by signal attenuation can derive the problem of data loss by the signal suppression effect. However, this problem can be avoided completely by assigning different channels to different groups. Lastly, to deal with external interference, the LBT mechanism can be employed. However, it is too costly if a task gives up sending data whenever it detects a channel is busy. Considering that external interference usually occurs transiently, we employ the *mLBT* mechanism in which a node repeats channel detection m times before it gives up transmission within the assigned slot.

III. REAL-TIME LoRa PROTOCOL

The real-time LoRa (*RT-LoRa*) protocol is explained formally with the design of a *logical slot indexing* (*LSI*) algorithm and a real-time task scheduling that utilizes the logical slot indices generated by the *LSI* algorithm.

A. FRAME STRUCTURE

Signals generated by different nodes experience different signal attenuations due to path loss and shadowing, it is not easy to avoid the signal suppression effect, even though two nodes are located at the same distance from GW. Another problem is that data transmissions using the same channel and SF are easily exposed to *collision* [33]. These two problems have to be considered in designing a new protocol. Since the proposed protocol targets a relatively small industrial field such as a manufacturing factory, nodes are divided into only two groups, G_L and G_H, that include nodes with relatively low and high signal attenuations, respectively. Then, two adjacent SFs are assigned to G_L and G_H such that nodes in G_H use the higher SF to better overcome high signal attenuation, and two different channels are assigned to G_L and G_H to avoid the signal suppression effect.

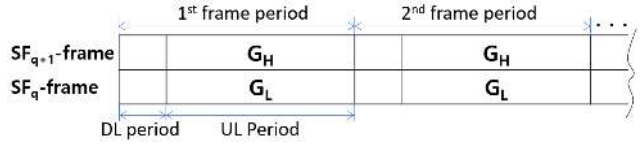


FIGURE 3. Frame structure using two SFs.

Two spreading factors SF_q and SF_{q+1} are used to define two frame types, SF_q-*frame* and SF_{q+1}-*frame*, that are associated with G_L and G_H, respectively where q ranges between 7 and 11. Fig. 3 illustrates a frame structure that consists of SF_q-*frame* and SF_{q+1}-*frame*. Each frame period consists of a downlink (*DL*) period for a DL message transmission by a server and an uplink (*UL*) period for data transmission by end nodes. During the DL period, nodes synchronize their clock times with a received DL message.

B. FRAME-SLOT STRUCTURE

Since the packet ToA using SF_{q+1} is approximately twice as long as the one using SF_q, if SF_q-*frame* is divided into 2^N slots, SF_{q+1}-*frame* is divided into 2^{N-1} slots approximately. Each frame can be reused with additional channels, referred to as a *channel-assisted frame reuse*. Fig. 4 shows a frame-slot structure when k channels are used for the SF_q-*frame* type and m-k channels are used for the SF_{q+1}-*frame* type. In this case, the total number of slots, *nSlots*, is given as follows:

$$nSlots = \left(\frac{(m+k)}{2}\right) \times 2^N \tag{2}$$

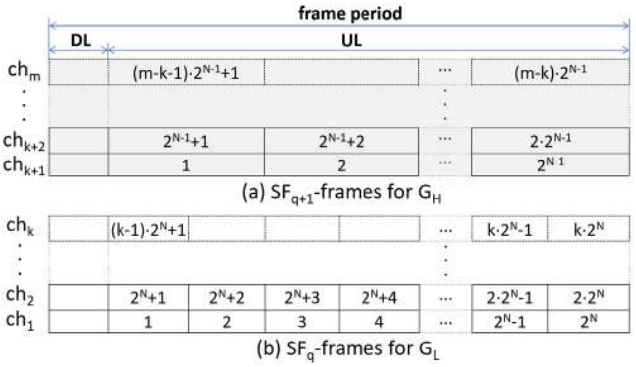


FIGURE 4. A group-based frame-slot structure.

For example, if $m = 6$, $k = 3$, and $N = 6$, $nSlots = 288$. With $q = 7$, suppose that one UL slot is 100 ms long and one DL slot is 200 ms long. If every node sends one packet per a frame period (= 6.6 s), 288 nodes can send packets to GW during the frame period.

C. LOGICAL SLOT INDEXING

A *logical slot indexing (LSI)* algorithm assigns a *logical index* to each slot in a frame such that given a frame of 2^N slots, if a task with $P = {}^{2^N}/_{2^k}$ selects 2^k slots sequentially starting from an arbitrary logical index, it can send one data packet per period P to a server. For example, suppose that any two sequential logical indices are assigned to two slots such that they do not belong to the same side of two equally divided parts of the frame. Then, if a task with period $P = {}^{2^N}/_{2^1}$ slots selects those two slots, it can send one packet per P. This principle has to hold for any task with various periods.

Definition 1: Given any logical index i, j sequential logical slot indices from i to $i + j - 1$ are said to be *sound* if $2^{k-1} < j \leq 2^k$ and each of them is assigned to a slot that belongs to one of the 2^k equally divided sections of a frame.

Definition 2: The j sequential logical slot indices from 1 to j are said to be *feasible* if (1) the $j-1$ sequential logical slot indices from 1 to $j-1$ are *feasible* and (2) for all $i < k$ such that $2^{k-1} \leq j < 2^k$, the logical slot indices from $j - 2^i + 1$ to j are sound.

Definition 2 defines the feasibility of the sequential logical slot indices in a recursive form where condition (2) guarantees the soundness of 2^i logical slot indices backward from j, recursively such that (j), $(j-1, j)$, $(j-3, j-2, j-1, j)$, $\ldots$, and $(j - 2^i + 1, \ldots, j-1, j)$ are sound. However, this does not guarantee that $(j-2, j-1)$ are sound, thereby requiring condition (1).

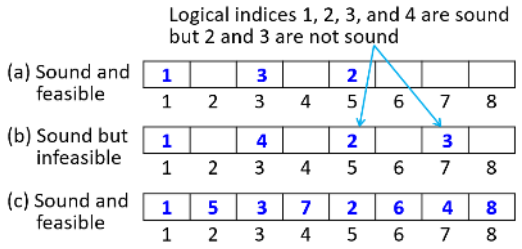


FIGURE 5. An example of soundness and feasibility of the sequential logical slot indices.

Take a look at Fig. 5(a). According to Definition 1, the three assigned logical slot indices (1, 2, 3) are sound since j (= 3) falls in $(2^1, 2^2]$ and each logical index belongs to one of 2^2 equally divided sections. According to Definition 2, they are feasible since when $j = 2$, (1, 2) is feasible and when $j = 3$, (2, 3) is recursively sound such that (3) and (2, 3) are sound. However, the logical slot indices (1, 2, 3, 4) in Fig. 5(b) are not feasible since condition (1) of Definition 2 is satisfied such that (1, 2, 3) is not feasible since (2, 3) is not sound. Fig. 5(c) shows an example of a feasible sequence.

The design of the LSI algorithm follows Definition 2. The algorithm can guarantee condition (1) by repeating "*satisfying the recursive soundness*" given in condition (2) with the whole frame whenever it assigns a new logical index. To realize condition (2) in assigning a new index j, the algorithm selects the section with the smaller maximum index after dividing the whole section into two smaller sections, thereby making $(j-1, j)$ sound. If it performs this process one more time with the selected section, it can make $(j-3, j-2, j-1, j)$ sound. It can continue this process to guarantee the recursive soundness until it finds an index-free section or finds that the whole frame is fully indexed. The algorithm is detailed in Algorithm 1.

Let us take an example to see how the LSI algorithm works with a frame of $N = 3$. Without loss of generality, assume that the algorithm always selects the first slot within an index-free section to assign a new logical index. The algorithm starts with $lsi = 1$ at line 1. Then, the outer *while loop* in line 2 is repeated until the frame is fully indexed. First, the selected section becomes the whole frame $S^0(0)$ (line 3). Since the selected section is index-free, the logical index 1 is assigned to the first slot in the frame after the first round of the outer while loop. Next, the algorithm repeats the outer while loop to assign the logical slot index 2. The selected section, *selectedS*, becomes the whole frame $S^0(1)$ in line 3. Since it is not index-free, it is equally divided into two smaller sections, $S^1(1)$ with the first four slots and $S^1(0)$ with the later four slots (line 6). Then, the algorithm selects $S^1(0)$ since $0 < 1$ (line 8). Since the selected section is index-free, the algorithm exits the inner while loop (line 5). Then, lsi(= 2) is assigned to the first slot of the selected section $S^1(0)$ (line 14), resulting in $S^1(2)$. Since the whole frame $S^0(2)$ is not fully indexed, the algorithm continues the outer while loop to assign the next lsi(= 3). The final indexing result is given in Fig. 6.

Algorithm 1 Logical Slot Indexing (LSI)

Input: A given *index-free* frame that has 2^N slots
Output: The frame with the feasible logical slot indices
\# *lsi*: A logical slot index
\# *selectedS*: A variable to select a section
\# $S^i(\alpha)$: The i$^{\text{th}}$ divided section whose max. logical index is α

1: $lsi \leftarrow 1$
2: **while** $S^0(lsi-1)$ is not fully indexed **do**
3: $selectedS \leftarrow S^0(lsi-1)$ # the whole frame
4: $i \leftarrow 1$
5: **while** *selectedS* is not index-free **do**
6: divide *selectedS* into $S^i(u)$ and $S^i(v)$
7: **if** $u < v$ **then**
8: $selectedS \leftarrow S^i(u)$
9: **else**
10: $selectedS \leftarrow S^i(v)$
11: **end if**
12: $i \leftarrow i+1$
13: **end while**
14: assign *lsi* to one arbitrary slot in *selectedS*
15: $lsi \leftarrow lsi+1$
16: **end while**

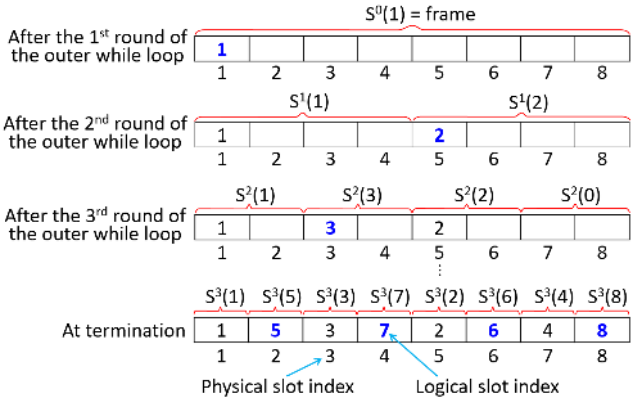


FIGURE 6. Execution of the LSI algorithm to assign the logical slot indices to a frame of 8 slots.

Given a frame of n slots, the outer while loop is repeated n times, and the inner while loop is executed $\log n$ times since the search space is reduced by half. In addition, in each iteration i of the inner while loop, $n/2^i$ comparisons for each of two divided sections are made to find the maximum slot index. Therefore, the time complexity function $T(n)$ of this algorithm can be expressed as follows: $T(n) = n\sum_{i=1}^{\log n} \frac{2n}{2^i}$. Therefore, $T(n) \in O(n^2)$. Note that the algorithm is executed once only if the frame size is redefined.

Lemma 1: Suppose that 2^k sequential logical indices are assigned and are feasible. Then, if the next logical index $2^k + 1$ is assigned by Algorithm 1, the logical slot indices $(2, 3, \ldots, 2^k, 2^k + 1)$ are recursively sound.

Proof: Let us see how a logical slot index $2^k + 1$ is assigned by Algorithm 1, referring to Fig. 7. It starts with the

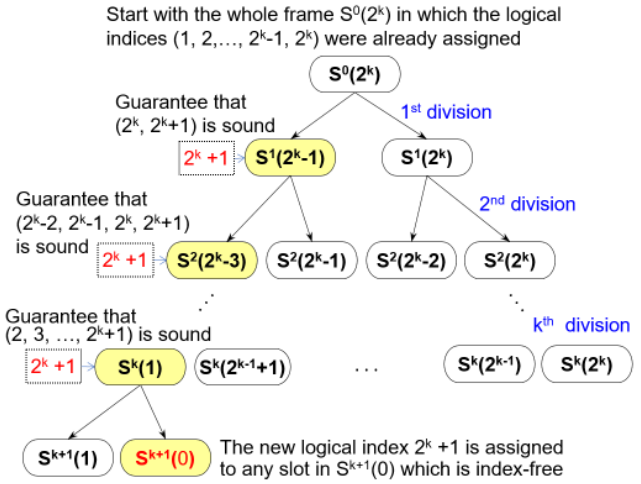


FIGURE 7. A process of assigning a new logical index $2^k + 1$ (the value in ( ) indicates the maximum logical index when the corresponding section is examined).

whole frame $S^0\left(2^k\right)$ in which 2^k sequential logical indices $\left(1, 2, \ldots, 2^k - 1, 2^k\right)$ were already assigned. Since $S^0\left(2^k\right)$ is not index-free, it starts the inner while loop. It divides $S^0\left(2^k\right)$ into $S^1\left(2^k - 1\right)$ and $S^1\left(2^k\right)$ in line 6. This corresponds to 1st division in Fig. 7. Then, it selects section $S^1\left(2^k - 1\right)$ that has the smaller maximum logical index to assign $2^k + 1$. This guarantees that $\left(2^k, 2^k + 1\right)$ is sound. However, if $S^1\left(2^k - 1\right)$ is not index-free, it has to be further divided into $S^2\left(2^k - 3\right)$ and $S^2\left(2^k - 1\right)$, resulting in the selection of $S^2\left(2^k - 3\right)$ to assign $2^k + 1$, thereby making $\left(2^k - 2, 2^k - 1, 2^k, 2^k + 1\right)$ sound (2^{nd} division). The algorithm repeats the inner loop until it finds an index-free section. Continuing this process, the algorithm will assign $2^k + 1$ to any empty slot in $S^k\left(2^k - \left(2^k - 1\right)\right)$ (or $S^k\left(1\right)$) (k^{th} division), thereby making $\left(2, 3, \ldots, 2^k, 2^k + 1\right)$ sound. Thus, we prove Lemma 1.

Lemma 2: If logical slot indices are assigned by Algorithm 1, 2^i sequential logical indices from j are feasible.

Proof: When $i = 1$, we prove that 2^1 logical indices, j and $j + 1$ assigned according to Algorithm 1 are feasible. Assume that the logical index j was already assigned to any empty slot in the frame by Algorithm 1. Then, the whole frame is $S^0\left(j\right)$. To assign the logical index $j+1$, the algorithm divides $S^0\left(j\right)$ into two smaller sections, $S^1\left(j - 1\right)$ and $S^1\left(j\right)$ and tries to assign $j + 1$ to $S^1\left(j - 1\right)$ that has the smaller maximum logical index. This guarantees that $(j, j + 1)$ are sound. Since j alone is feasible and $(j, j + 1)$ is recursively sound, $(j, j + 1)$ is feasible by Definition 2.

When $i = k$, we assume that 2^k logical indices from j to $j + 2^k - 1$ are feasible. When $i = k + 1$, we prove that 2^{k+1} sequential logical indices starting with j are feasible.

By assumption, the sequence $(j, j + 1, \ldots, j + 2^k - 1)$ is feasible. Therefore, by Lemma 1, $(j + 1, j + 2, \ldots, j + 2^k - 1, j+2^k)$ is recursively sound. Since $\left(j, j + 1, \ldots, j + 2^k - 1\right)$ is feasible and $\left(j + 1, j + 2, \ldots, j + 2^k - 1, j + 2^k\right)$ is recursively sound, $(j, j + 1, \ldots, j + 2^k - 1, j + 2^k)$ is feasible by Definition 2. In the same way, $(j, j + 1, \ldots, j + 2^k - 1, j + 2^k, j + 2^k + 1)$ is feasible. Continuing this, 2^{k+1} logical indices $(j, j + 1, \ldots, j + 2^k - 1, j + 2^k, \ldots, j + 2^{k+1} - 1)$ become feasible. Thus, we prove Lemma 2.

Theorem 1: Consider a frame of 2^N slots logically indexed by Algorithm 1. Given a task $\tau_i = (P_i), P_i = {}^{2^N}\!/_{2^k}$ slots, $(0 \le k \le N)$, task τ_i can meet its deadline if it takes slots with 2^k sequential logical indices starting with any logical index.

Proof: Suppose that 2^k sequential slots from j to $j+2^k-1$ are selected. Then, by Lemma 2, the selected logical indices are feasible. This implies that the task is assigned one slot in each section of ${}^{2^N}\!/_{2^k}$ slots. Thus, if the task sends one data packet in each of the selected slots, the packet can be transmitted within its period P_i or before its deadline that becomes the start of next period. Thus, we prove Theorem 1.

Lemma 3: Given two overlapping frames, the first frame is logically indexed from 1 to 2^N and the second frame is logically indexed from $2^N + 1$ to 2^{N+1} by Algorithm 1, they can be treated as a concatenated frame in scheduling under the constraint that a task can select at most 2^N slots.

Proof: Suppose that the first 2^N logical slot indices in the first frame are denoted as $a_1, a_2, \ldots, a_{2^N}$ and the last 2^N logical indices in the second frame are denoted as $b_1, b_2, \ldots, b_{2^N}$.

Suppose that 2^k $(0 \le k \le N)$ sequential logical slots are selected such that $x\left(x \le 2^k\right)$ slots $(a_{2^N-x+1}, a_{2^N-x+2}, \ldots, a_{2^N})$ belong to the first frame and next $2^k - x$ slots $(b_1, b_2, \ldots, b_{2^k-x})$ belong to the second frame. We need to prove two things: (1) none of the selected slots is overlapped with any others and (2) they are sound. Since a_i is overlapped with b_i, $2^k - x$ selected slots on the second frame are equivalent to $(a_1, a_2, \ldots, a_{2^k-x})$ on the first frame by replacing b_i by a_i. They become the combination of $(a_1, a_2, \ldots, a_{2^k-x})$ and $(a_{2^N-x+1}, a_{2^N-x+2}, \ldots, a_{2^N})$ on the first frame. Since $2^k - x < 2^N - x + 1$ with any k $(0 \le k \le N)$, the selected slots never overlap with any others. Thus, (1) is true.

As in Lemma 1, the k^{th} division guarantees that $(a_1, a_2, \ldots, a_{2^k})$ is sound and a_{i+2^k} is assigned to the same section with a_i. Thus, $(a_{i+1}, a_{i+2}, \ldots, a_{i+2^k})$ is also sound on the same sections in which $(a_1, a_2, \ldots, a_{2^k})$ is sound. Consequently, if $i = 2^N - 2^k$, $(a_{2^N-2^k+1}, a_{2^N-2^k+2}, \ldots, a_{2^N})$ is sound on the same sections that $(a_1, a_2, \ldots, a_{2^k})$ is sound. Therefore, the combination of selected slots $(a_1, a_2, \ldots, a_{2^k-x})$ and $(a_{2^N-x+1}, a_{2^N-x+2}, \ldots, a_{2^N})$ are also sound at the same sections since $x < 2^k$. Thus, (2) is true and we prove Lemma 3.

Theorem 2: Suppose that k frames are logically indexed from 1 to $k \cdot 2^N$ by Algorithm 1. Given a task, $\tau_i = (P_i)$, $P_i = {}^{2^N}\!/_{2^k}$ slots, $(0 \le k \le N)$, task τ_i can meet its deadline if it takes the slots with 2^k sequential logical indices from any logical index.

Proof: By Lemma 3, k frames can be treated as one concatenated frame in task scheduling. Thus, 2^k selected logical indices are sound. This implies that the task τ_i is assigned one slot in each section of ${}^{2^N}\!/_{2^k}$ slots and the packet can be transmitted within its deadline. Thus, we prove Theorem 2.

D. GROUPING AND SCHEDULING

1) NODE GROUPING

A *level-based grouping* method is introduced. Every node determines its level as the hop distance to GW that reflects the degree of signal attenuation when it receives a message. After the construction of level topology, the nodes in the same level belong to the same group.

To construct level topology, GW broadcasts a *group request*(*GR*) message, $GR = (lvl = 0)$, using SF_q. Upon receiving *GR*, if the SNR is greater than or equal to a specified threshold value, a node determines its *level* as $lvl + 1$ and retransmits $GR = (1)$ immediately to increase the effect of concurrent transmission [26]. Upon receiving $GR = (1)$, every node sets its *level* to 2 $(= lvl + 1)$. Then, every node with $level = 1$ belongs to G_L while one with $level = 2$ belongs to G_H. In this paper, since one GW can cover only the nodes of up to two hops, other nodes that fail to receive $GR = (1)$ can be covered by another GW.

2) TASK SCHEDULING SCHEME

According to Theorem 2, given k concatenated frames, any task $\tau_i = (P_i)$ is simply scheduled by selecting ${}^{2^N}/_{P_i}\left(\leq 2^N\right)$ slots sequentially starting with any logical slot index. Let Γ denote a list of tasks for a group: $\Gamma = (\tau_1, \tau_2, \ldots, \tau_n)$. A scheduler can schedule task τ_i as long as the concatenated frame of $k \cdot 2^N$ slots has idle slots corresponding to ${}^{2^N}/_{P_i}$ sequential logical slot indices. The starting logical slot index becomes one plus the last logical index whose slot was allocated to task τ_{i-1}.

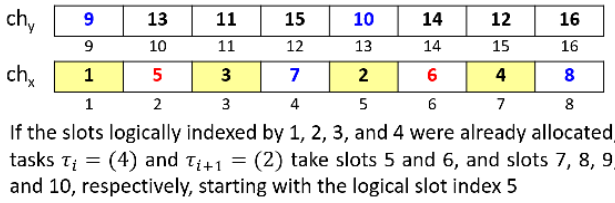


FIGURE 8. Task scheduling example with two concatenated frames of N = 3.

For example, consider two concatenated frames of $N = 3$ in Fig. 8. Suppose that the logical indices 1, 2, 3, and 4 were already allocated to a partial list of tasks, $(\tau_1, \tau_2, \ldots, \tau_{i-1})$. Then, the following tasks $\tau_i = (4)$ and $\tau_{i+1} = (2)$ take the logical indices from 5 to 6 and the logical indices from 7 to 10, respectively.

A task set Γ is schedulable with k concatenated frames if the total number of slots required by tasks in Γ does not exceed the number of slots defined in the k frames. Thus, a task set Γ is schedulable if and only if it satisfies the following inequality:

$$\frac{\sum_{i=1}^{n} {}^{2^N}/_{P_i}}{k \times 2^N} \leq 1 \tag{3}$$

TABLE 1. An example of a task schedule.

Task ID	Task class i	Number of assigned slots, 2^i	Assigned logical slot indices (*startIndex* = 5)
A	2	4	5 to 8
B	1	2	9 to 10
C	1	2	11 to 12
D	0	1	13
E	0	1	14

In (3), the numerator indicates the total number of slots required by n tasks in task set Γ and the denominator does the total number of available slots in the k concatenated frames. Therefore, if the numerator is less than and equal to the denominator, it is obvious that every task can acquire its required slots.

3) GROUP SCHEDULING INFORMATION AND DISTRIBUTION

Tasks are grouped into task classes such that if a task has the transmission period of ${}^{2^N}/_{2^c}$, it belongs to task class c, where c ranges in $[0, maxClass]$, $maxClass \leq N$. Then, a task τ_i that belongs to class c is expressed as follows:

$$\tau_i = (c) \tag{4}$$

Then, a *group scheduling information* for group G_x, *GSI(x),* is expressed as follows:

$$GSI(x) = (x, startIndex, T_c, T_{c-1}, \ldots, T_i, \ldots, T_1, T_0)$$

where *startIndex* is the *start logical index* in the scheduling of G_x, and T_i is a set of tasks in class i, given as follows:

$$T_i = (i, k_i, (\tau_{i1}, \tau_{i2}, \ldots, \tau_{ij}, \ldots, \tau_{ik_i}))$$

where i, k_i, and τ_{ij} indicate task class i, the number of tasks in class i, and the j^{th} task in task class i, respectively. A server can easily maintain the addition of new tasks by distributing a partial *GSI* only for the new tasks.

Let us see an example for task scheduling. Consider group x that has five tasks such as task A in class 2, tasks B and C in class 1, and tasks D and E in class 0. If $startIndex = 5$, $GSI(x)$ is represented as follows:

$$GSI(x) = (x, 5, (2, 1, (A)), (1, 2, (B, C)), (0, 2, (D, E)))$$

When a server distributes $GSI(x)$, every node can generate the same task schedule as given in Table 1.

E. COMMAND AND DATA TRANSMISSION

Every task sends data to GW in each of the physical slots that correspond to the assigned logical slot indices. For example, in Table 1, task B sends data in the physical slots that correspond to the logical slot indices 9 and 10.

A server can send a control packet, *CTRL*, to end nodes during the DL period. *CTRL* can include a control packet header (*CPH*) and three optional elements in brackets:

$$CTRL = (CPH, [GSI(x)], [CMD], [NACK])$$

where *CPH* = (*Network ID*, *frame period*) is used to inform a node of the network to which it belongs, *frame period* indicates the period number of the current frame, *CMD* is a command, and *NACK* (*negative acknowledgement*) includes the list of nodes that failed to deliver data for a specified number of frame periods. *GSI(x)* can be fragmented into multiple smaller *GSIs* and transmitted over multiple frame periods. Upon receiving *NACK*, a node can know whether its link to GW is reliable or not.

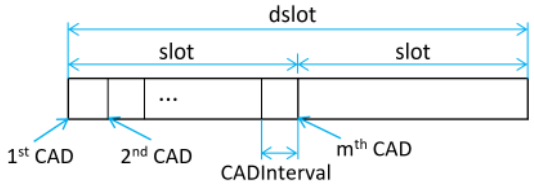


FIGURE 9. Operation of the mLBT mechanism with a double sized slot.

Another issue is external interference. A node tends to experience external interference transiently. Thus, it is not desirable for the node to give up data transmission immediately when it detects the channel is busy. An *mLBT* mechanism is introduced that allows a node to perform channel activity detection (CAD) $m\,(\geq 2)$ times. A slot length is extended to the length of double slots, named *dslot* as shown in Fig. 9. The interval of CAD, *CADInterval*, can be set as follows.

$$CADInterval = \frac{slot}{m-1}, \quad m \geq 2 \tag{5}$$

If $m = 1$, the *mLBT* mechanism degenerates to the *LBT* one.

IV. PERFORMANCE EVALUATION

A. THE LEAST UPPER BOUND OF NETWORK THROUGHPUT

In the Poisson process, given a packet arrival rate λ, $P(k)$, the probability that k packets will arrive in time t, is given as:

$$P(k \text{ attempts in time } t) = \frac{(\lambda t)^k}{k!} e^{-\lambda t} \tag{6}$$

Let T_P denote one packet transmission time that is assumed to be equal to a slot time in this analysis. Suppose that each node transmits one packet during a frame period, T_F. Then, this transmission model can be transformed into the Poisson process with the mean probability of T_P/T_F. Then, for n nodes, the packet arrival rate λ can be expressed as follows:

$$\lambda = \frac{nT_P}{T_F} \tag{7}$$

Since LoRaWAN follows the Aloha protocol in data transmission, the vulnerable periods of a data packet using LoRaWAN and Slotted Aloha correspond to $2T_P$ and T_P, respectively [13]. Let $P_0(X)$ denote the probability that a packet is transmitted successfully to a server for protocol X. By applying the average arrival rate in (7) to (6), we get $P_0(X)$ as follows:

$$P_0(LoRaWAN) = P(k=0 \text{ in } 2 \text{ time slots}) = e^{-\frac{2nT_P}{T_F}} \tag{8}$$

$$P_0(Slotted\ Aloha) = P(k=0 \text{ in } 1 \text{ time slot}) = e^{-\frac{nT_P}{T_F}} \tag{9}$$

Since there are λ transmission attempts in one frame period, the throughput $S(X)$ of protocol X is calculated as follows:

$$\lambda S(LoRaWAN) = \lambda \cdot P_0(LoRaWAN) = \frac{nT_P}{T_F} \cdot e^{-\frac{2nT_P}{T_F}} \tag{10}$$

$$\lambda S(SlottedAloha) = \lambda \cdot P_0(SlottedLoRa) = \frac{nT_P}{T_F} \cdot e^{-\frac{nT_P}{T_F}} \tag{11}$$

Let us calculate the throughputs of *RT-LoRa(LBT)* and *RT-LoRa(mLBT)* that indicate RT-LoRa with *LBT* and *mLBT*, respectively. The size of a downlink slot is very short compared with the whole frame period; it can be omitted in the calculation. Therefore, with *RT-LoRa(LBT)*, the maximum number of slots per a frame is T_F/T_P. Since a slot schedule is used, one packet is successfully transmitted per one slot. Thus, the throughput of *RT-LoRa(LBT)*, $S(RT-LoRa(LBT))$, is given as follows:

$$S(RT-LoRa(LBT)) = \begin{cases} \frac{nT_p}{T_F}, & \text{if } n \leq \frac{T_F}{T_P} \\ 1, & \text{if } n > \frac{T_F}{T_P} \end{cases} \tag{12}$$

In this formula, the throughput increases linearly until n reaches the maximum number of slots. Once it reaches the maximum of *unity*, it is sustained since additional transmission is not allowed. With *RT-LoRa(mLBT)*, the maximum number of slots per a frame is $T_F/2T_P$ since the size of a slot becomes doubled. In consequence, the throughput of RT-LoRa(mLBT), $S(RT-LoRa(mLBT))$, is simply changed as follows:

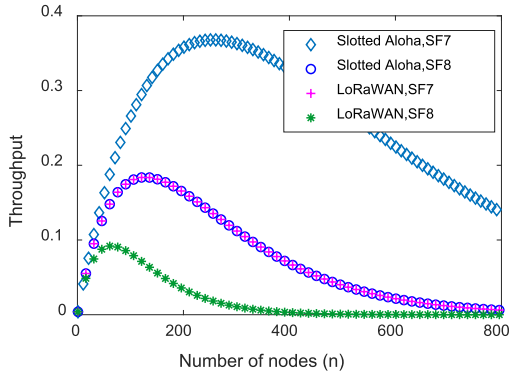
$$S(RT-LoRa(mLBT)) = \begin{cases} \frac{nT_p}{T_F}, & \text{if } n \leq \frac{T_F}{2T_P} \\ \frac{1}{2}, & \text{if } n > \frac{T_F}{2T_P} \end{cases} \tag{13}$$

With this approach, the maximum throughput is reduced by half since only one packet can be transmitted within the double sized slot, *dslot*.

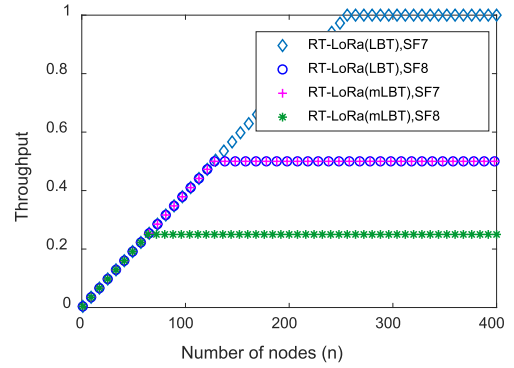
Fig. 10 shows throughputs for LoRaWAN, Slotted Aloha, RT-LoRa(LBT), and RT-LoRa(mLBT) from (10), (11), (12), and (13) when SF_7 and SF_8 are used. Each graph shows the least upper bound of throughput with $T_P = 100\ ms$ and $T_F = 25.6\ s$. With these values, the frame period can be divided into 2^8 uplink slots with SF_7 or 2^7 uplink slots with SF_8.

Referring to Fig. 10(a), Slotted Aloha shows much higher least upper bound in throughput than LoRaWAN since its vulnerable period is reduced by half. Similarly, Slotted Aloha using SF_7 increases the least upper bound greatly over Slotted Aloha using SF_8.

In Fig. 10(b), the throughputs of RT-LoRa(LBT) and RT-LoRa(mLBT) show a linearly increasing pattern as the



(a) LoRaWAN and Slotted Aloha



(b) RT-LoRa

FIGURE 10. The least upper bounds of throughput for different protocols with $T_P = 100$ ms and $T_F = 25.6$ s.

TABLE 2. Experimental parameters and values.

Parameter	Value	Parameter	Value
Number of nodes	15	Spreading factors	SF_7
Packet size	33 bytes	Bandwidth	125 kHz
Number of channels	1	Code rate	4/5
Tx power	14 dBm	Preamble	8 symbols

number of transmitting nodes increases until a frame is saturated. Since the number of slots with SF_7 is twice as many as that with SF_8, the graph of RT-LoRa(LBT) using SF_8 becomes saturated when the number of nodes reaches 2^7. Note that RT-LoRa(LBT) with SF_7 achieves the maximum throughput of 1.0 (in Fig. 10(b)) while Slotted Aloha with SF_7 lowers the maximum throughput down to 0.36 (in Fig. 10(a)), with 256 nodes.

B. EFFECT OF INTERFERENCE

1) EXPERIMENTAL SETUP AND SCENARIOS

Experiment was performed in the testbed of one GW and fifteen nodes with STM32 microcontroller and SX1276 radio chip. It is claimed that the use of one channel and one SF is sufficient to examine the advantages of the protocols comparatively. Nodes were deployed over one three-floor building at the University of Ulsan. Three protocols, *LoRaWAN*, *Slotted Aloha*, and *RT-LoRa*, were compared for their PDRs. The key parameters and values used in the experiments are summarized in Table 2.

The following two scenarios were used for experiment.

Scenario 1 with no external interference
Each node transmits one packet to GW with Tx intervals of 1.5 s, 3 s, 6 s, or 12 s without any scheduled external interference.
Scenario 2 with external interference
Each node transmits one packet to GW every 3 seconds on average. Additional 5 interfering nodes transmit garbage packets at Tx intervals of 1.5 s, 3 s, 6 s, or 12 s.

Since this experiment is concerned with internal and external interference, SF_7 only was used and every node was checked beforehand to make sure that its link to GW is good.

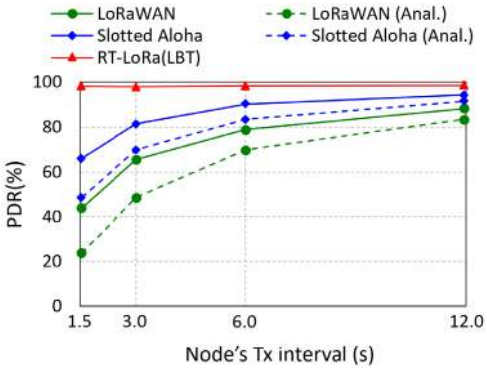


FIGURE 11. Comparison of PDRs for different protocols with varying node Tx interval without interfering nodes.

2) EXPERIMENTAL RESULTS

Experiment was performed with Scenario 1 to evaluate the effect of internal interference. Fig. 11 compares PDRs for three protocols. It is shown that RT-LoRa achieves PDR of 100% while LoRaWAN and Slotted Aloha decrease PDR sharply as traffic increases. Note that the PDR of LoRaWAN decreases down to 44% when every node has Tx interval of 1.5 s. The figure also examines the soundness of the experimental data by comparing them with the analytical data from (8) for LoRaWAN and (9) for Slotted Aloha with $T_P = 72\ ms$ and $T_F = Txinterval$. Two dashed curves indicate the analytical data for LoRaWAN and Slotted Aloha, respectively. Analytical graphs for LoWaWAN and Slotted Aloha show their PDRs less than their experimental data by 45% and 27%, respectively at the high traffic of $Txinterval = 1.5$. This is because the analytical model does not take into account capture effect. According to one study [14], it was shown that PDRs by analytical model are lower than ones by simulation that reflects capture effect by up to 40% depending on traffic. However, note that they have similar decreasing patterns.

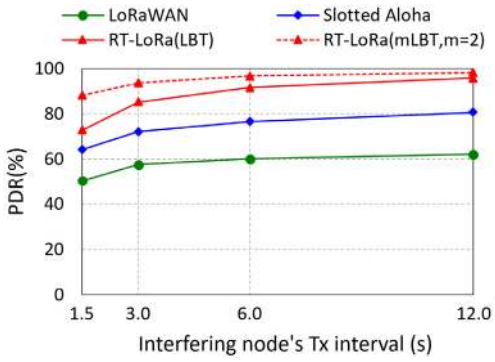


FIGURE 12. Comparison of PDRs for different protocols with varying interfering node Tx interval (node Tx interval is fixed to 3 s).

Experiment was performed with Scenario 2 to evaluate the effect of external interference. Fig. 12 shows the PDRs of LoRaWAN, Slotted Aloha, RT-LoRa(LBT) and RT-LoRa(mLBT) according to the change of external interference with the node's Tx interval fixed to 3 s. RT-LoRa(mLBT, m = 2) achieves higher PDR than Slotted Aloha by 20% overall, and shows high dependability against the increase of interfering signal. The results verify that the use of *mLBT* is effective even with $m = 2$, achieving PDR of almost 100% with Tx interval of interfering nodes over 6 s. Note that LoRaWAN only achieves the low PDR of around 60% for such external interfering traffic load.

C. EFFECT OF GROUPING

First, experiment was performed to evaluate the effect of signal suppression and collision in data transmission. Second, experiment was performed to evaluate the effect of grouping in RT-LoRa against signal suppression by comparing with two other protocols, LoRaWAN and Slotted Aloha. To increase the effectiveness of experiment, in Table 2, the number of nodes and the packet size were increased to 20 and 38, respectively and two spreading factors, SF_7 and SF_8 were used for two node groups.

Using the level-based grouping method, a two-level topology was constructed with 20 nodes deployed across the University campus buildings, thereby producing two groups, G_L of 13 nodes and G_H of 7 nodes. All nodes in G_L use SF_7 to achieve high throughput and those in G_H use SF_8 to overcome the signal attenuation problem. Every end node is requested to send one packet per 3 seconds on average.

First, two data transmission approaches, *slot scheduled transmission* and *random transmission* were examined by enabling and disabling the operation of the nodes in G_L. Note that nodes in G_H are affected by signal suppression when the nodes in G_L are enabled to send data. PDRs were calculated only for G_H nodes, and appear as three bar graphs in Fig. 13. Fig. 13(a) and (b) show the PDRs of the slot scheduled transmission approach and the random transmission one when G_L nodes are disabled. The slot scheduled transmission approach achieves the PDR of 94% with the small number of packets lost by only signal attenuation. However, with the random transmission approach, the PDR decreases down to 68%

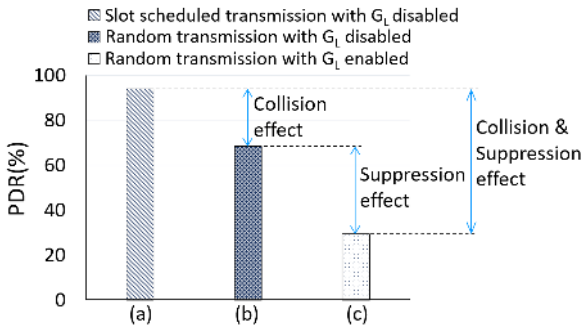


FIGURE 13. Examination on the effect of collision and signal suppression for the nodes in G_H.

due to both signal attenuation and data collision. Fig. 13(c) shows the PDR of the random transmission approach when G_L nodes are enabled. In this case, nodes suffer from both collision and signal suppression, resulting in the significant degradation in PDR down to 30%.

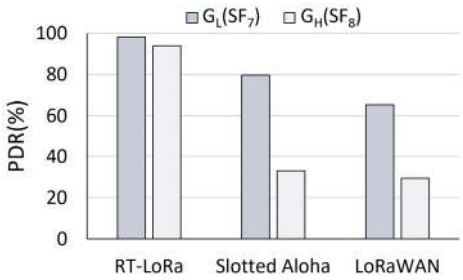


FIGURE 14. Examination on the effect of signal suppression and data collision for different protocols.

Second, experiment was performed to examine the effect of grouping with the RT-LoRa protocol. In this experiment, since RT-LoRa prevents signal suppression in the design of the frame-slot architecture, it was tested for G_H and G_L separately. Fig. 14 shows two PDR bar graphs for each protocol, one for group G_L and another for G_H. RT-LoRa shows PDR of 98% for G_L and PDR of 94% for G_H since it can avoid both collision and signal suppression. It is shown that the PDRs of LoRaWAN decrease to 65% for G_L due to collision and 29% for G_H due to both collision and signal suppression. These experimental results are in well accord with the ones obtained with G_L nodes enabled in the random transmission approach. Slotted Aloha could improve PDRs slightly, by 5% for G_H and 15% for G_L, for the same scenario.

V. CONCLUSION

A real-time LoRa protocol was proposed for the use in industrial monitoring and control applications. It used a real-time task scheduling algorithm, supported by the logical slot indexing algorithm. The logical slot indices enable the efficient generation of a schedule for real-time tasks. To overcome signal attenuation and signal suppression effect, node grouping method was proposed, and its effectiveness was proven by experiment. The RT-LoRa protocol reinforced by the *mLBT* mechanism (even with $m = 2$) was examined on the testbed of one gateway and fifteen test nodes in campus. Even with quite high traffic that each node generates one

packet per 3 seconds on average, it achieved the PDR over 94% in spite of high external interference such that each of five interfering nodes generates garbage packet per 3 seconds. According to our experiment, it was found that the nodes deployed in wireless unfriendly zones could not reach GW directly. In further study, this RT-LoRa protocol will be extended to enable data transmission via multiple wireless hops.

REFERENCES

[1] L. Kay Soon, W. N. N. Win, and E. Meng Joo, "Wireless sensor networks for industrial environments," in *Proc. Int. Conf. Comput. Intell. Modelling, Control Automat. Int. Conf. Intell. Agents, Web Technol. Internet Commerce (CIMCA-IAWTIC)*, vol. 2, Nov. 2005, pp. 271–276, doi: 10.1109/CIMCA.2005.1631480.

[2] J. Heo, J. Hong, and Y. Cho, "EARQ: Energy aware routing for real-time and reliable communication in wireless industrial sensor networks," *IEEE Trans Ind. Informat.*, vol. 5, no. 1, pp. 3–11, Feb. 2009, doi: 10.1109/TII.2008.2011052.

[3] C. Dombrowski and J. Gross, "EchoRing: A low-latency, reliable token-passing mac protocol for wireless industrial networks," in *Proc. Eur. Wireless 21th Eur. Wireless Conf.*, May 2015, pp. 1–8.

[4] H. Oh and C. T. Ngo, "A slotted sense multiple access protocol for timely and reliable data transmission in dynamic wireless sensor networks," *IEEE Sensors J.*, vol. 18, no. 5, pp. 2184–2194, Mar. 2018, doi: 10.1109/JSEN.2018.2790422.

[5] P. Suriyachai, J. Brown, and U. Roedig, "Time-critical data delivery in wireless sensor networks," in *Distributed Computing in Sensor Systems*, R. Rajaraman, T. Moscibroda, A. Dunkels, A. Scaglione, Eds. Berlin, Germany: Springer, 2010, pp. 216–229.

[6] R. Sanchez-Iborra and M.-D. Cano, "State of the art in LP-WAN solutions for industrial IoT services," *Sensors*, vol. 16, no. 5, p. 708, May 2016. [Online]. Available: https://www.mdpi.com/1424-8220/16/5/708

[7] M. Luvisotto, F. Tramarin, L. Vangelista, and S. Vitturi, "On the use of LoRaWAN for indoor industrial IoT applications," *Wireless Commun. Mobile Comput.*, vol. 2018, pp. 1–11, Art. no. 3982646, doi: 10.1155/2018/3982646.

[8] D. Chen, M. Nixon, and A. Mok, *WirelessHARTTM: Real-Time Mesh Network for Industrial Automation*. New York, NY, USA: Springer, 2010, p. 276.

[9] S. Petersen and S. Carlsen, "WirelessHART versus ISA100.11a: The format war hits the factory floor," *IEEE Ind. Electron. Mag.*, vol. 5, no. 4, pp. 23–34, Dec. 2011, doi: 10.1109/MIE.2011.943023.

[10] D. De Guglielmo, S. Brienza, and G. Anastasi, "IEEE 802.15.4e: A survey," *Comput. Commun.*, vol. 88, pp. 1–24, Aug. 2016, doi: 10.1016/j.comcom.2016.05.004.

[11] L. Vangelista, A. Zanella, and M. Zorzi, "Long-range IoT technologies: The dawn of LoRa," in *Future Access Enablers for Ubiquitous and Intelligent Infrastructures*, V. Atanasovski and A. Leon-Garcia, Eds. Cham, Switzerland: Springer, 2015, pp. 51–58.

[12] M. A. Ertürk, M. A. Aydın, M. T. Büyükakka lar, and H. Evirgen, "A survey on LoRaWAN architecture, protocol and technologies," *Future Internet*, vol. 11, no. 10, p. 216, Oct. 2019. [Online]. Available: https://www.mdpi.com/1999-5903/11/10/216

[13] A. Brand and H. Aghvami, *Multiple Access Protocols for Mobile Communications: GPRS, UMTS and Beyond*. New York, NY, USA: Wiley, 2002.

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

[15] 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, doi: 10.1109/LWC.2016.2647247.

[16] K. Mikhaylov, P. Juha, and T. Haenninen, "Analysis of capacity and scalability of the LoRa low power wide area network technology," in *Proc. Eur. Wireless 22th Eur. Wireless Conf.*, May 2016, pp. 1–6.

[17] 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, doi: 10.1109/JIOT.2017.2768498.

[18] 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, doi: 10.3390/s19040838.

[19] M. Bor, J. Vidler, and U. Roedig, "LoRa for the Internet of Things," presented at the Proc. Int. Conf. Embedded Wireless Syst. Netw., Graz, Austria, 2016.

[20] J. P. Queralta, T. N. Gia, Z. Zou, H. Tenhunen, T. Westerlund, "Comparative study of LPWAN technologies on unlicensed bands for M2M communication in the IoT: Beyond LoRa and LoRaWAN," *Procedia Comput. Sci.*, vol. 155, pp. 343–350, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050919309639, doi: 10.1016/j.procs.2019.08.049.

[21] A. Bachir, M. Dohler, T. Watteyne, and K. K. Leung, "MAC essentials for wireless sensor networks," *IEEE Commun. Surveys Tuts.*, vol. 12, no. 2, pp. 222–248, 2nd Quart., 2010, doi: 10.1109/SURV.2010.020510.00058.

[22] A. S. Althobaiti and M. Abdullah, "Medium access control protocols for wireless sensor networks classifications and cross-layering," *Procedia Comput. Sci.*, vol. 65, pp. 4–16, Jan. 2015, doi: 10.1016/j.procs.2015.09.070.

[23] 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, doi: 10.1109/ACCESS.2019.2913985.

[24] 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, doi: 10.3390/s18113718.

[25] 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, doi: 10.1109/ACCESS.2017.2755858.

[26] 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.

[27] 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, doi: 10.1109/WFCS.2017.7991972.

[28] D. Croce, M. Gucciardo, S. Mangione, G. Santaromita, and I. Tinnirello, "Impact of LoRa imperfect orthogonality: Analysis of link-level performance," *IEEE Commun. Lett.*, vol. 22, no. 4, pp. 796–799, Apr. 2018, doi: 10.1109/LCOMM.2018.2797057.

[29] L. Leonardi, F. Battaglia, G. Patti, and L. L. Bello, "Industrial LoRa: A novel medium access strategy for LoRa in industry 4.0 applications," in *Proc. 44th Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Oct. 2018, pp. 4141–4146, doi: 10.1109/IECON.2018.8591568.

[30] D. Zorbas and B. O'Flynn, "Autonomous collision-free scheduling for LoRa-based industrial Internet of Things," in *Proc. IEEE 20th Int. Symp. 'World Wireless, Mobile Multimedia Netw.' (WoWMoM)*, Jun. 2019, pp. 1–5, doi: 10.1109/WoWMoM.2019.8792975.

[31] 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, doi: 10.1109/ETFA.2017.8247601.

[32] L. Tessaro, C. Raffaldi, M. Rossi, and D. Brunelli, "Lightweight synchronization algorithm with self-calibration for industrial LORA sensor networks," in *Proc. Workshop Metrol. Ind. 4.0 (IoT)*, Apr. 2018, pp. 259–263, doi: 10.1109/METROI4.2018.8428309.

[33] J. Haxhibeqiri, F. Van den Abeele, I. Moerman, and J. Hoebeke, "LoRa scalability: A simulation model based on interference measurements," *Sensors*, vol. 17, no. 6, p. 1193, May 2017. [Online]. Available: https://www.mdpi.com/1424-8220/17/6/1193.

QUY LAM HOANG received the B.S. degree from the School of Electrical Engineering, Hanoi University of Science and Technology, Vietnam, in 2015. 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 received the dual B.S. degrees in electrical and computer engineering and in 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.

DAESEUNG 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 Sungkyunkwan University, Seoul, South Korea, and the M.Sc. and Ph.D. degrees in computer science from Texas A&M University at College Station, Texas, in 1993 and 1995, respectively. From 1983 to 1989 and from 1996 to 2000, he worked as a Software Engineer and as a Software Architect with the Corporate Research Center of Samsung Electronics. He was involved in developing communication protocols for the data services of the CDMA and IMT2000 handset products. 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, South Korea, where he has been serving as the Department Head of the IT Convergence Department, since 2016. He received the Best Paper Award from the National Academy of Science, USA, in 1995. He published over 50 refereed journals for the last decade and made many technology transfers to the local IT companies through University–Industry Cooperation Program. His research interests lie in embedded systems, mobile ad hoc networks, real-time computing, and context-aware computing. He is a member of IEICE and ICASE, and also a Lifetime Member of KICS and KISA.