This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2022.3162019, IEEE Internet of Things Journal

A Real-Time LoRa Protocol Using Logical Frame Partitioning for Periodic and Aperiodic Data Transmission

Quy Lam Hoang and Hoon Oh, *Member, IEEE*

***Abstract*—Owing to provision of a long-range and robust link, LoRa technology has drawn attention for the use in industrial data collection networks. This paper proposes a real-time LoRa protocol that can effectively deal with both periodic and aperiodic data. The slots in a frame are logically partitioned such that they are first scheduled for periodic data by a slot scheduling algorithm, and then, the remaining unscheduled slots are used for event-driven aperiodic data. In this logical frame partitioning, the unscheduled slots appear in an interleaved fashion so that aperiodic tasks can transmit data with low delay and fairness while every periodic task still completes data transmission before the beginning of the next period. To deal with the problems of data collision and traffic congestion for aperiodic data, a two-level collision avoidance scheme is proposed that adopts the notion of a contention window and a delay slot. According to simulation, not only can the proposed protocol guarantee the timely delivery of periodic data, but it can also deal with aperiodic data with high reliability, fairness, and low delay, compared with other recent protocols.**

***Index Terms*—LoRa protocol, frame partitioning, periodic data, aperiodic data, CSMA/CA, contention window, listen-before-talk.**

I. Introduction

Recently, industrial applications have been considering the use of LoRa technology [1] for data collection networks owing to the provision of a long-range, stable link and ease of deployment [2, 3]. One of those applications is an industrial safety monitoring and control system in which a server collects data from end devices or nodes and provides a safety service based on analysis of the collected data. The server may collect data from a node periodically to keep track of the location and condition of workers or equipment, or to monitor time-varying situations in a target work field. On the other hand, a server might collect data in an event-driven manner. For example, the server might ask a specific node to turn a specific sensor module in the node on or off for energy management. Then, the node will change the operating status of the sensor module and will report the result to the server. As another example, a node might have to transmit data *instantly* if it detects abnormal or emergency situations, such as emission of toxic gas or a worker's entrance into a restricted access zone. This requires the design of a LoRa MAC protocol that can effectively deal with both periodic and aperiodic data.

Many studies regarding reliable transmission for two types of data have been conducted so far under wireless communication technologies such as WiFi and Bluetooth. The WiFi protocol pursues a best-effort service at different quality of service (QoS) levels [4] using two modes: the distributed coordination function (DCF) and the point coordination function (PCF). In the former, WiFi employs the carrier sense multiple access with collision avoidance (CSMA/CA) mechanism for reliable data transmission. The latter mode uses a superframe that consists of a *contention period* (CP) and a *contention-free period* (CFP), and employs CSMA/CA to avoid collisions during CP, and uses slot scheduling during CFP for guaranteed transmission. Bluetooth also supports different QoS levels in best-effort or guaranteed best–effort transmission [5]; however, it incurs some additional overhead by having a master node control transmissions of a slave node for reliability. Nevertheless, neither WiFi nor Bluetooth will experience noticeable degradation of performance, because they have high bandwidth and use a tiny control message compared to the data.

In wireless sensor networks (WSNs) [6] or LPWA networks [7] (such as LoRa, SigFox, and NB-IoT with restricted resources and low data rates), control messages should be used in a limited manner with careful consideration. Similar to WiFi, the IEEE 802.15.4 standard [8] uses a superframe that consists of a *contention access period* (CAP) and a *contention-free period* (CFP); but it incurs problems such as unbounded delay and limited reliability in data transmission, according to some studies [9, 10]. The IEEE 802.15.4e standard [11], approved in 2012, addressed those problems. It defines different MAC behavior modes to deal with various requirements for industrial applications. Among them, the Deterministic and Synchronous Multi-channel Extension (DSME) mode was introduced to achieve deterministic delay and high reliability in data transmissions and to improve adaptability to time-varying traffic. Even though DSME mode provides the mechanisms to accommodate both periodic and aperiodic data, it does not provide a complete implementation that can deal with a dynamic network topology in industrial environments [11]. In fact, a change

This work was supported by Institute of Information & communication Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (2020-0-00869, Development of 5G-based Shipbuilding & Marine Smart Communication Platform and Convergence Service). (*Corresponding author: Hoon Oh.*)

The authors are with the Department of Electrical, Electronic and Computer Engineering, University of Ulsan, Ulsan 44610, South Korea (email: quylam925@gmail.com, hoonoh@ulsan.ac.kr).

in topology can not only damage the quality of a real-time service, but also increases control overhead considerably by triggering network reconstruction and slot rescheduling.

Even though WiFi, Bluetooth, and WSNs provide the mechanisms for reliable transmission of periodic and aperiodic data, WiFi and Bluetooth are seldom used in data collection networks due to limited scalability, spectrum inefficiency, or high energy consumption, and WSNs have difficulty in meeting industrial requirements due to multi-hop routing and the dynamic nature of the WSN topology [12]. LoRa technology can overcome those limitations owing to a simple network topology enabled by its long transmission range and low energy consumption, a capture effect that enables protective operation from other LoRa networks [13], and provision of high link stability and multiple data rates [14, 15].

A lot of effort was made to secure reliability of data transmission in LoRa networks. A well-known LoRa protocol is LoRaWAN [16], designed for low data–rate applications, where nodes transmit data using the Aloha protocol, and can receive a downlink message from a server in one of two receive windows that are reserved after each of two specified time intervals. The early studies regarding data transmission mostly focused on evaluating the performance of LoRaWAN [17-20], indicating that LoRaWAN is not suitable for applications that have relatively heavy traffic due to data collisions. Some studies tried to improve reliability in data transmission by using slotted transmission [21, 22] or CSMA with the listen-before-talk (LBT) mechanism [23, 24]. Some other studies dealt with the problem of reliable and/or real-time data transmission for industrial applications [25-27]. In [25], the authors mentioned the adaptation of the time-slotted channel hopping (TSCH) [11] (used in IEEE 802.15.4e) to LoRa networks for reliability, but without giving a specific slot scheduling method. In [26], Zorbas and Flynn proposed distributed slot scheduling using a hash function with which every node determines its own slots so different nodes do not occupy the same slot. With this approach, wasting slots is inevitable since the frame size should be large enough to guarantee distinct allocation of slots. In [27], the real-time LoRa protocol generates a slot schedule based on the data transmission periods of nodes such that every node can transmit data before the beginning of the next period if it sends data according to the slot schedule; but it does not address transmission of aperiodic data.

Meanwhile, a couple of studies dealt with both periodic and aperiodic data in LoRa networks. Leonardi et al. proposed the Industrial LoRa protocol (ILoRa) based on the frame division approach (used in the IEEE 802.15.4 standard) that physically splits a superframe [28]. They used the Aloha protocol during CAP for aperiodic data transmission while using an offline slot schedule during CFP. Later, they proposed RT-LoRa by improving LoRa to support a real-time service [29]. The RT-LoRa protocol uses Slotted Aloha during CAP for aperiodic data transmission while it suggests a slot scheduling approach during CFP for periodic data transmission. For a real-time service, it defines a superframe that corresponds to the shortest data transmission period among nodes and that requires every node to transmit one data every superframe to meet time constraints. However, a node that has a transmission period corresponding to multiple superframes can waste some slots since it occupies multiple slots instead of just one slot. They also presented a QoS-oriented slot scheduling mechanism to meet different QoS requirements for periodic data. That mechanism classifies nodes into one of three QoS classes (*Normal*, *Reliable*, and *Most Reliable*) and assigns resources, such as spreading factors (SFs) and transmission slots, differently to the nodes according to the QoS level. Specifically, a *Normal* node, a *Reliable* node, and a *Most Reliable* node, respectively, are assigned one slot associated with the lowest possible SF, one slot associated with the highest possible SF, and multiple slots, each associated with a distinct SF. RT-LoRa tried to support both real-time service and QoS together; however, it can increase the size of a superframe and considerably restrict schedulability because a high SF occupies a long slot. Furthermore, two nodes using different SFs on the same channel can collide due to imperfect orthogonality if their signal strength offset is less than a specified threshold [30].

This paper presents a novel real-time LoRa MAC protocol to efficiently support two types of data. The protocol uses *logical frame partitioning* such that the slots in the frame are logically divided by the slot scheduling algorithm into a set of scheduled slots for periodic data and a set of unscheduled slots for aperiodic data. In this partitioning, the scheduled slots and the unscheduled slots appear in interleaved fashion instead of in two physically split sections. This allows a node with aperiodic data to quickly find an unscheduled slot. To secure reliability in aperiodic data transmissions, the proposed protocol includes a *two-level collision avoidance* scheme in which a node selects an unscheduled slot among a specified number of unscheduled slots, named a *contention window*, and then it contends for a channel in the selected slot again using a random number of *delay slots* to avoid collisions before data transmission. The former and latter schemes are respectively called the *first-level collision avoidance* (*L1-CA*) and the *second-level collision avoidance* (*L2-CA*). If a node fails to acquire a channel in the selected slot, it restarts L1-CA with a doubly increased contention window for congestion control.

The proposed protocol was compared with ILoRa [28] and RT-LoRa [29] in terms of packet delivery ratio (PDR), and delay and fairness in data transmissions under three scenarios differentiated by node distribution, traffic load, and intensity of the hidden node problem. Simulation showed that the proposed protocol could far outperform the others in reliability and average transmission delay, and could greatly improve on the unfairness of data transmissions caused by the capture effect. Furthermore, simulation results showed that the use of multiple gateways (GWs) could alleviate the hidden node problem considerably.

II. Preliminaries

A. Network Model

The considered LoRa network consists of one server, multiple gateways, and a number of end devices or nodes. A node can communicate directly with at least one of the GWs that connect to the LoRa server via a high-speed backhaul network. Every node has one *periodic task* that transmits data to a GW, and can activate an *aperiodic task* to transmit data when an event occurs. The periodic task has to transmit data before the beginning of the next period or deadline. Different tasks can have different transmission periods. If a server receives identical data multiple times via different GWs, it accepts only one of them.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2022.3162019, IEEE Internet of Things Journal

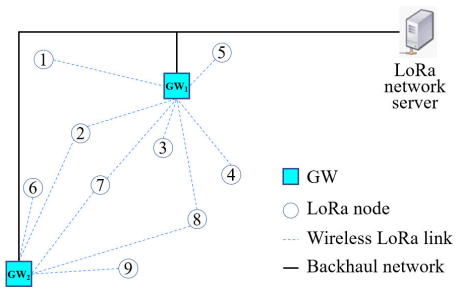


Fig. 1. The LoRa network model

Fig. 1 shows a simple LoRa network that includes one network server, two gateways (GW_1 and GW_2), and nine nodes. Three nodes 2, 7, and 8 are covered by both GW_1 and GW_2 connected to the server, whereas the other nodes are covered by only one GW.

B. Notations

For convenience, τ_x denotes a *periodic task* on node x, represented as follows:

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

where p_x indicates a node address and a transmission period, and a_x denotes an *aperiodic task* that arrives at time t on node x, represented as follows:

$$a_x = (x, t)$$

We assume that all aperiodic data can be transmitted within one data slot. If the data cannot fit into one data slot, they can be segmented and transmitted separately. Table I summarizes important notations used in this paper.

TABLE I
SUMMARY OF NOTATIONS.

Notation	Meaning
Ch_i-*frame*	A frame defined on channel i
N	A frame factor used in defining a frame of 2^N slots
G_i	A group of periodic tasks that are scheduled on Ch_i-*frame*
$nSch_i$	The number of scheduled slots on Ch_i-*frame*
$nUSch_i$	The number of unscheduled slots on Ch_i-*frame*
T_i	A set of periodic tasks that have a transmission period of 2^i slots
$sLSI$	A start logical slot index to schedule new tasks
GSI	Group scheduling information, $GSI = (g, sLSI, T_0, T_1, \ldots, T_N)$ where g indicates group number
$uTSI$	Updated task scheduling information, $uTSI = (i, (x, sLSI))$ where i and x indicate group number and task identification number, respectively
$d(\tau)$	Slot demand of periodic task τ per frame period
$D(T_i)$	Slot demand of T_i per frame period
$preD(\tau)$	Total slot demand of all periodic tasks that precede task τ in GSI
psi	Physical slot index as the index of an array
$SS(\tau)$	Slot schedule of periodic task τ that is expressed as a set of physical slot indices
CW_i	Contention window defined by an aperiodic task after the i^{th} failure to get a channel
CW_i^e	Extended contention window corresponding to CW_i

C. Scheduling of Periodic Tasks

In this subsection, slot scheduling for periodic tasks, as presented in [27], is briefly introduced since its result determines the slots for data transmission of aperiodic tasks. For easy slot scheduling of periodic tasks, the authors presented the *logical slot indexing* (*LSI*) algorithm, which assigns a logical slot index to every slot in a frame of 2^N slots such that if a task with a transmission period of 2^{N-k} slots is assigned 2^k slots corresponding to the 2^k sequential logical slot indices starting with any logical slot index, and transmits one data in each assigned slot, it can always transmit one data per its transmission period.

For an easy explanation of the LSI algorithm, we introduce the notion of 2^k-*Constraint*. Given a frame of 2^N slots, it is said that 2^k logical indices satisfy the 2^k-*Constraint* if only one of them appears in each of the 2^k equally divided sections of the frame. The LSI algorithm using 2^k-*Constraint* is given in Algorithm 1.

Algorithm 1. Logical slot indexing

// *Assume that a frame has* 2^N *slots*
1. assign logical indices 1 and 2 to the first half and the second half of the frame, respectively;
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, from the statement in line 1, it is obvious that the first two logical indices satisfy the 2^1-*Constraint*. When $i = 2$ (line 2), logical indices 3 and 4 are assigned (line 3) such that when 3 is assigned, (3, 2) satisfies the 2^1-*Constraint*, and when 4 is assigned, (4, 3) satisfies the 2^1-*Constraint*, and (4, 3, 2, 1) satisfies the 2^2-*Constraint* in a recursive manner (line 4). Similarly, when $i = 3$, logical indices 5, 6, 7, and 8 are assigned sequentially such that each assignment satisfies the corresponding constraint.

D. Motivation

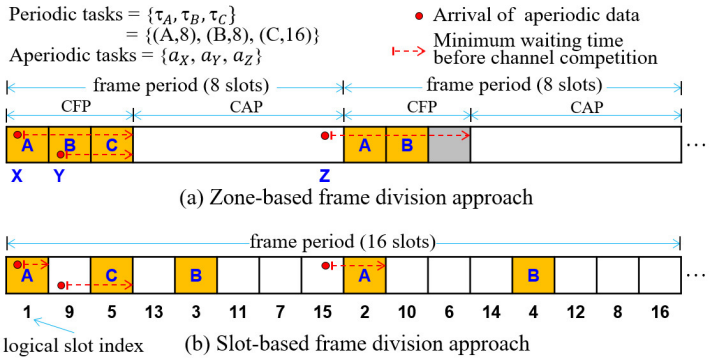


Fig. 2. Comparison of the two frame division approaches

The RT-LoRa protocol [29] uses the same frame division approach as the IEEE 802.15.4 standard, hereafter referred to as *zone-based frame division (ZFD)*. In ZFD, since a frame size is determined as the smallest of all the task periods, every periodic task has its transmission period as the integer multiple of the frame. Therefore, a task can waste $(n - 1)$ slots if it has transmission period $n \times frame$, $n \geq 1$. Furthermore, if an aperiodic task arrives during the CFP, it has to wait until the beginning of CAP.

According to the discussion of the ZFD approach, it suffers from a wasting of slots and a transmission delay for aperiodic data. These problems can be tackled by using slot scheduling for the periodic tasks. Given a set of periodic tasks, each of them is scheduled such

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2022.3162019, IEEE Internet of Things Journal

that it obtains slots corresponding to sequential logical slot indices starting with the last scheduled logical slot index. After scheduling all the periodic tasks, the unscheduled slots are accessed through contention among the aperiodic tasks. In this way, the scheduled slots and the unscheduled slots appear in interleaved fashion, hereafter referred to as *slot-based frame division (SFD)*. Therefore, the SFD approach can reduce the average transmission delay of an aperiodic task, compared to the ZFD approach.

Fig. 2 illustrates how periodic and aperiodic tasks are scheduled with ZFD and SFD. Consider a set of periodic tasks, $\{\tau_A, \tau_B, \tau_C\} = \{(A, 8), (B, 8), (C, 16)\}$, and three aperiodic tasks, a_X, a_Y, and a_Z, with their respective arrival times as shown in the figure. Referring to Fig. 2-(a), which illustrates scheduling in ZFD, the frame size is determined based on a periodic task with a minimum period (i.e., eight slots). Two periodic tasks, τ_A and τ_B, are assigned one slot per frame, while task τ_C gets only one slot per two frame periods, resulting in one wasted slot. Fig. 2-(b) illustrates scheduling in SFD. The frame size is determined to be sufficiently large by considering the total slot demand of periodic tasks and their periods. In this example, the frame size is 16 slots long. Periodic tasks τ_A, τ_B, and τ_C are assigned five slots, starting with a logical slot index of 1. Observe that non-deterministically arriving aperiodic tasks can quickly get an unscheduled slot.

The use of SFD gives the following advantages. First, the exact number of required slots can be allocated to periodic tasks by using the slot scheduling algorithm without wasting slots. Second, the distributed unscheduled slots allow an aperiodic task to find an unscheduled slot quickly. In ZFD, for channel contention, an aperiodic task has to wait until the start of the next CAP unless it arrives during CAP; or, if it arrives in the latter part of CAP, it might fail to transmit data, and must wait for the whole CFP. Third, since aperiodic tasks contend for a channel only at the slot boundary, it naturally takes advantage of the principle of Slotted Aloha. Finally, aperiodic tasks achieve fairness in data transmission due to the randomness of their arrivals and the distribution of unscheduled slots.

III. Real-Time LoRa Protocol with Logical Frame Partitioning

The *real-time LoRa protocol with logical frame partitioning* (*RTLoRa-LFP*) is explained with the frame structure design and a method of slot scheduling that utilizes the logical slot indices. Then, the *two-level collision avoidance* scheme is presented for the reliable data transmission of aperiodic tasks.

A. Frame-Slot Architecture

With *m* available channels, *m* overlapping frames are defined for each frame period, and each frame consists of a downlink (DL) segment and an uplink (UL) segment. A frame using channel Ch_i is denoted by *Ch$_i$-frame.* The server uses the DL segment or a DL slot to transmit a downlink message to nodes, whereas nodes use the UL segment to send periodic or aperiodic data. The UL segment is further sliced into 2^N *data slots* where *N* as a *frame factor* is an integer constant, and each data slot is sufficiently large to send one data packet. Each frame is identified by a distinct channel, and the data slots in the UL segment of each frame are identified by *physical slot indices*, numbered sequentially from 1 to 2^N, and each of the frames is used for the scheduling of tasks independently.

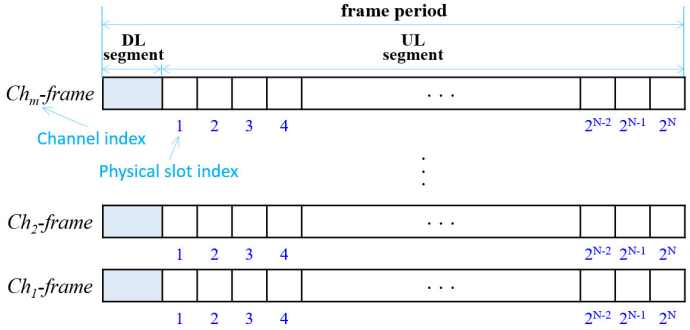


Fig. 3. The multi-channel frame-slot architecture

A server uses one specified channel to broadcast a DL message during the DL segment, while all nodes have to wait for the message on the channel. In this paper, we assume that every channel uses one distinct SF to prevent collisions by imperfect orthogonality [30]. Note that channel hopping can be easily used to improve the transmission reliability of the DL message. A frame-slot architecture using *m* channels is illustrated in Fig. 3.

B. Logical Frame Partitioning

Apart from the physical slot indices, the data slots in each of the frames are assigned the same logical slot indices by the LSI algorithm. With *m* channels, all periodic tasks are divided into *m* groups, $G_1, G_2, \ldots, G_m$, and $G_i = (\tau_{i,1}, \tau_{i,2}, \ldots, \tau_{i,n_i})$, where $\tau_{i,j}$ is the j^{th} task of group G_i, and n_i is the number of periodic tasks in G_i. If all periodic tasks in group G_i are scheduled, $nSch_i$ as the total number of scheduled slots for G_i, is given as follows:

$$nSch_i = \sum_{j=1}^{n_i} d(\tau_{i,j}) \le 2^N \quad (1)$$

where $d(\tau_{i,j})$ is the slot demand of task $\tau_{i,j}$ per frame period. Then, $nUSch_i$, as the total number of unscheduled slots in *Ch$_i$-frame*, is given: $nUSch_i = 2^N - nSch_i$.

By the slot scheduling of periodic tasks, the data slots in *Ch$_i$-frame* are said to be *logically partitioned* such that $nSch_i$ data slots corresponding to the logical slot numbers from 1 to $nSch_i$ are scheduled, and $nUSch_i$ data slots from $nSch_i + 1$ to 2^N are unscheduled. Note that $nSch_i$ implies the last scheduled slot in *Ch$_i$-frame*, and this way of partitioning the slots is referred to as *logical frame partitioning*.

C. Slot Scheduling

Since tasks in one group are independent of tasks in other groups, slot scheduling of periodic tasks for all groups is identical. Thus, we describe scheduling for only one group in this subsection. A server generates *group scheduling information* (GSI) for slot scheduling of a group as follows:

$$GSI = (g, sLSI, T_0, T_1, \ldots, T_N)$$

where g is the group number, *sLSI* indicates the *start logical slot index* to start generation of a slot schedule for the tasks, and T_i clusters periodic tasks that have the same transmission period of 2^i slots just for the reduction of the size of GSI. Thus, T_i is represented as follows:

$$T_i = (\tau_{i,1}, \tau_{i,2}, \ldots, \tau_{i,n_i})$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2022.3162019, IEEE Internet of Things Journal

where $\tau_{i,j}$ is the j^{th} task of T_i, and n_i is the number of tasks in T_i. Note that *sLSI* is always *unity* since the slot scheduling of every group starts with an empty schedule at the beginning.

For slot scheduling of periodic tasks, the server distributes the GSI. Upon receiving the GSI, if a task finds itself in group g, it generates its slot schedule. $SS(\tau_{i,j})$ as the slot schedule for task $\tau_{i,j}$ can be obtained as follows:

$$SS(\tau_{i,j}) = \begin{pmatrix} psi(preD(\tau_{i,j}) + sLSI), psi(preD(\tau_{i,j}) + sLSI + 1), \\ \dots, psi(preD(\tau_{i,j}) + sLSI + d(\tau_{i,j}) - 1) \end{pmatrix} \quad (2)$$

where $psi(x)$ indicates the translation of the logical slot index x to the *physical slot index*, and $preD(\tau_{i,j})$ indicates the total slot demand of all tasks preceding task $\tau_{i,j}$ in GSI as follows:

$$preD(\tau_{i,j}) = \sum_{k=0}^{i-1} D(T_k) + \sum_{k=1}^{j-1} d(\tau_{i,k}) \quad (3)$$

where $D(T_k)$ indicates the total slot demand of all tasks in T_k.

If the size of GSI exceeds the size of the data that the server can transmit in one DL slot, GSI can be divided into k subgroups, $sG(1)$, $sG(2)$, …, $sG(k)$ with $GSI(1), GSI(2), \dots, GSI(k)$, $k > 1$, respectively, to be transmitted separately. In this case, the server calculates $sLSI(i)$ as the start logical slot index of subgroup i, and includes it in $GSI(i)$ before sending,

$$sLSI(i) = \sum_{j=1}^{i-1} D(sG(j)) \quad (4)$$

and

$$GSI(i) = (g, sLSI(i), T_0, T_1, \dots, T_N)$$

where $D(sG(j))$ indicates the total slot demand of subgroup i. Upon receiving $GSI(i)$, a task can generate its slot schedule by using (2) after replacing *sLSI* with $sLSI(i)$.

To analyze the time complexity of $SS(\tau_{i,j})$ in (2), let n_{max} be the maximum number of tasks that a group can have, and d_{max} be the maximum slot demand that a task can have in the group. Note that $n_{max} = d_{max} = 2^N$, where 2^N is the number of slots in a frame. Also note that if a group has 2^N tasks, $d_{max} = 1$. $SS(\tau_{i,j})$ performs $preD(\tau_{i,j})$ only once, $d(\tau_{i,j})$ additions, and for each term, calculates $psi(x)$. Since $preD(\tau_{i,j})$ requires $(n_{max} - 1)$ additions in the worst case and $psi(x)$ is $O(1)$ if it uses the conversion table of 2^N array, the time complexity of $SS(\tau_{i,j})$ is $d_{max} + n_{max} - 1 \in O(2^N)$. It is worth noting that $d(\tau_{i,j})$ is typically a small constant in real applications.

D. Data Transmission and Maintenance

1) Periodic Data Transmission

During network initialization, a node registers with a server by sending its task information (for x, $\tau_x = (x, p_x)$) and waits for a DL message on the specified channel to receive scheduling information from the server. If the server broadcasts GSI, every task in the group generates its own slot schedule in the corresponding frame and transmits data according to the slot schedule.

As time progresses, a node can leave the network. Then, the server marks the slots scheduled to that node as unscheduled, thereby producing the fragmentation of a scheduled frame. Suppose that a new periodic task, say x, with slot demand $d(x)$ registers with the server later. Then, the server searches for the unscheduled slots corresponding to $d(x)$ sequential logical slot indices in any frame, say *Ch$_i$-frame*. The server first tries to find any fragmented slots that can satisfy the slot demand to reduce the fragmentation. If it fails, it probes the unscheduled slots from $nUSch_i + 1$. Let *sLSI* be the start of those $d(x)$ slots. Then, it generates the following updated task scheduling information:

$$uTSI = (i, (x, sLSI))$$

and broadcasts *uTSI* so that task x can generate its slot schedule, $SS(x)$, as follows:

$$SS(x) = (psi(sLSI), psi(sLSI + 1), \dots, psi(sLSI + d(x) - 1)) \quad (5)$$

Since $SS(x)$ performs $d(x)$ additions, its time complexity is $O(2^N)$. As previously mentioned in Subsection C, since $d(x)$ is usually a small constant, the time complexity is $O(1)$.

2) Aperiodic Data Transmission

Since an aperiodic task transmits data in unscheduled slots, it has to record the start logical slot index of unscheduled slots in the frame to which it belongs. For this, the server includes the logical frame partitioning information, $LFPi = (nSch_1, nSch_2, \dots, nSch_m)$, in a DL message so that an aperiodic task can record the last scheduled logical slot index in the frame to which it belongs.

An aperiodic task may have to compete with other aperiodic tasks within an unscheduled slot. In low-rate wireless networks, a collision is costly because it not only wastes precious bandwidth but it also increases network traffic due to retransmission. The IEEE 802.15.4 standard [8] deals with this problem by using the Slotted CSMA/CA mechanism based on an exponential backoff algorithm that uses the notion of *backoff period* as a delay unit (i.e., a slot corresponding to a time length of 20 symbols). Before transmitting a packet, every node takes $random(0, 2^{BE} - 1)$ backoff periods, where *BE* is the backoff exponent, and then performs Channel Clear Assessment (*CCA*) twice to improve the correctness of checking due to the presence of the interframe space. A node that detects a busy channel takes another random backoff by incrementing BE. This process is repeated a specified number of times.

The collision problem becomes more serious in extremely low-rate LoRa networks characterized by a long packet *time on air* (*ToA*). It can be more severe if more data slots have been scheduled for periodic tasks. Therefore, this requires a collision-avoidance scheme for the reliable data transmission of aperiodic tasks.

a) Contention Window and Delay Slot

Definition 1*. Given that a data slot is divided into a number of delay slots (*delayslots*) numbered sequentially,* delayslot *is the smallest time span, such that if two neighboring nodes ready for data transmission generate delay slot numbers by the difference of unity, and wait for* delayslots *corresponding to the delay slot number before sending data, the one with the larger number can overhear the signal issued by another.*

To effectively handle the collision problem in LoRa networks, this paper introduces the notion of *contention window* and *delayslot*. A *contention window* (CW) is a set of forthcoming unscheduled data slots determined after the arrival of an aperiodic task. The aperiodic

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2022.3162019, IEEE Internet of Things Journal

task doubles the CW size if it fails to acquire a channel within CW. $|CW_i|$ as the size of the contention window after the i^{th} failure to acquire a channel, is expressed as follows:

$$|CW_i| = \min\left(|CW_{i-1}| \times 2, MaxCW\right) \quad (6)$$

where *MaxCW* is the maximum size of a contention window.

Included in *delayslot* are the time to turn the radio chip on, the time to transfer a packet from a microprocessor to a radio chip buffer, and the time to perform *Channel Activity Detection* (CAD). The first two time durations are negligible in LoRa devices, compared to the last. Furthermore, according to a technical report from Semtech [31], experiments indicate that the CAD operation for only one symbol time can fail to correctly detect the status of a channel. Hence, they recommend CAD operation spanning the time for multiple symbols, depending on the spreading factors. Thus, $delayslot_i$ as the delay slot with the use of SF *i*, is given as follows:

$$delayslot_i = k_i \times SymTime_i \quad (7)$$

where k_i is a coefficient for SF *i*, and $SymTime_i$ is the symbol time (in milliseconds) when SF *i* is used. Table II shows the values of *delayslot* for different SFs.

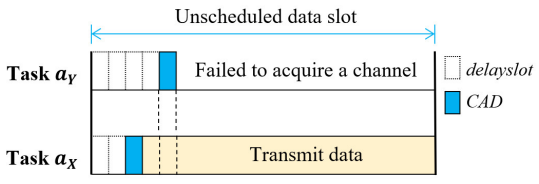
TABLE II
DELAYSLOT VALUES FOR DIFFERENT SFS (MS).

SF	7	8	9	10
k_i	2	2	4	4
$SymTime_i$	1.024	2.048	4.096	8.192
$delayslot_i$	2.048	4.096	16.384	32.768

BW=125kHz, CR=4/5

b) Two-Level Collision Avoidance Scheme

The *two-level collision avoidance* scheme is proposed based on contention window and *delayslot*. The *first-level collision avoidance* (*L1-CA*) scheme employs an exponential backoff algorithm that uses a data slot as a unit backoff. If an aperiodic task is activated, it first generates the initial contention window (CW_0) and randomly selects one data slot within CW_0. However, if multiple tasks select the same unscheduled slot, transmissions can result in a collision. Thus, an aperiodic task is asked to acquire a channel before sending data. This channel contention process is the *second-level collision avoidance* (*L2-CA*) scheme. Let us defer the discussion of L2-CA for the time being. Whenever a task fails to acquire a channel within the selected data slot, it defers transmission and doubles its contention window according to (6). A task gives up transmission after a specified number of failures. If it succeeds in sending, it changes its contention window back to CW_0.



Suppose that two aperiodic tasks a_X and a_Y selected the same unscheduled data slot but generated 2 and 4 *delayslots*, respectively. Task a_X wins the channel and transmits data while task a_Y defers its transmission.

Fig. 4. The operation of the L2-CA scheme

Now, let us see how the L2-CA scheme works. The L2-CA scheme uses the notion of *rdelay*, defined as follows:

$$rdelay = \text{random}\,(0, MaxDelayCnt) \times delayslot \quad (8)$$

where $MaxDelayCnt$ indicates a maximum delay count. To avoid collisions, every task takes delay for *rdelay* from the start of the selected data slot, and then performs CAD for one more delay slot. If the channel is idle, it sends data; otherwise, it defers data transmission and restarts L1-CA with the doubly increased contention window. Since the winning task has to complete data transmission within the remaining time of the selected slot, L2-CA does not use the backoff mechanism. Instead, it limits the maximum delay time within the data slot by using the constant *MaxDelayCnt*. If the size of aperiodic data is larger than the remaining time, a task segments the data to transmit in multiple slots. The segmented parts follow the same *two-level collision avoidance* scheme. The operation of L2-CA is illustrated in Fig. 4.

c) Extended Contention Window

Frames have different distributions of unscheduled slots if groups have different slot demands. An aperiodic task can transmit data on any channel. Once an aperiodic task arrives, CW is calculated as follows:

$$\begin{aligned} &\textit{Given aperiodic task } a_X = (X, t), \\ &S(a_X) = \{u | startT(u) \geq t, u \textit{ is an unscheduled slot}\} \\ &\textit{where } startT(u) \textit{ indicates the start time of slot } u. \end{aligned} \quad (9)$$

In (9), $S(a_X)$ includes all the unscheduled slots that appear after time *t* in all frames. When $|CW_0| = k$, CW_i of aperiodic task a_X after the i^{th} failure to get a channel is calculated as follows:

$$\begin{aligned} &CW_i \textit{ is a set of } (2^i \times k) \textit{ unscheduled slots such that} \\ &CW_i \subseteq S(a_X) \textit{ and } \forall u \in CW_i, \forall v \in (S(a_X) - CW_i)\,, \\ &startT(u) \leq startT(v). \end{aligned} \quad (10)$$

According to (10), a contention window may have to exclude some unscheduled slots that have the same start time, but that belong to different frames. However, it seems such exclusions are not desirable. Thus, CW_i^e, as an extended CW after a_X experiences the i^{th} failure, is calculated as follows:

$$\begin{aligned} &CW_i^e \textit{ of aperiodic task } a_X \textit{ is a set of unscheduled slots} \\ &\textit{such that } CW_i^e = \{u | u \in S(a_X), startT(u) \leq maxStartT\} \\ &\textit{where } maxStartT = max\{startT(u) | u \in CW_i\} \end{aligned} \quad (11)$$

Algorithm 2. Get extended contention window

```
// CH = a set of available channels
// CWsize = contention window size after the i^th failure
// firstpsi = the first physical slot index right after the i^th failure
// lsi(k) = logical slot index corresponding to physical slot index k
1. CW_i^e = ∅; psi = firstpsi;
2. while |CW_i^e| < CWsize do
3.     CW_i^e = CW_i^e ∪ {(c, psi) | lsi(psi) > nSch_c, c ∈ CH};
4.     psi = psi + 1;
5.     if psi > 2^N then psi = 1;
6. endwhile
```

In the implementation, CW_i^e can be expressed as a set of pairs (*channel* and *psi*), obtained as follows. If an aperiodic task arrives at time *t*, each unscheduled slot after time *t* is examined for each

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2022.3162019, IEEE Internet of Things Journal

channel *ch*, and (*ch*, *psi*) is included in CW_i^e if the logical slot index corresponding to *psi* is greater than the last scheduled logical slot index. This process repeats until the size of CW_i^e is greater than, or equal to, the current contention window size. This is given in Algorithm 2.

For example, consider task a_x that arrives at the down-arrowed time on node *x* as illustrated in Fig. 5. Let us calculate CW_0^e with a contention window size of 4. Fig. 5 illustrates two frames over two continuous frame periods. Each frame consists of eight slots, in which four logical slots (1, 2, 3, and 4) and two logical slots (1 and 2) are scheduled for periodic tasks in G_1 and G_2, respectively. Suppose that aperiodic task a_X arrives at time *t*. From Algorithm 2, *firstpsi* is 7, and $CW_0^e = \{(2,7), (1,8), (2,8), (1,2), (2,2)\}$.

If CW_0^e includes elements past one frame period, some elements can be duplicated. This problem can be easily resolved by including *frame period (fp)* in a tuple as (*fp*, *ch*, *psi*).

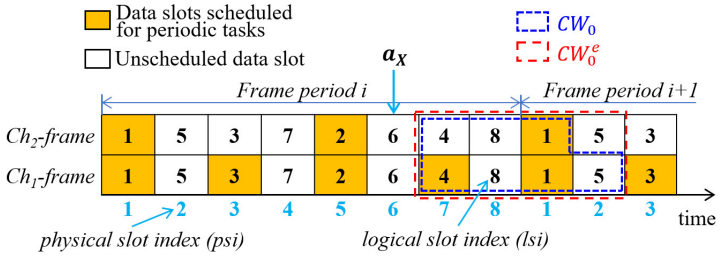


Fig. 5. Extended contention window for multichannel frames with an initial contention window size of 4

Fig. 6 illustrates the two-level collision avoidance scheme when two aperiodic tasks a_X and a_Y compete for data transmission. Suppose that tasks a_X and a_Y arrive at the beginning of the frame period. According to the L1-CA scheme, if they select different unscheduled data slots within CW_0^e, both will succeed in transmission. Even though they happen to select the same unscheduled data slot (say, the third one in the figure), they have another chance to avoid a collision by using the L2-CA scheme.

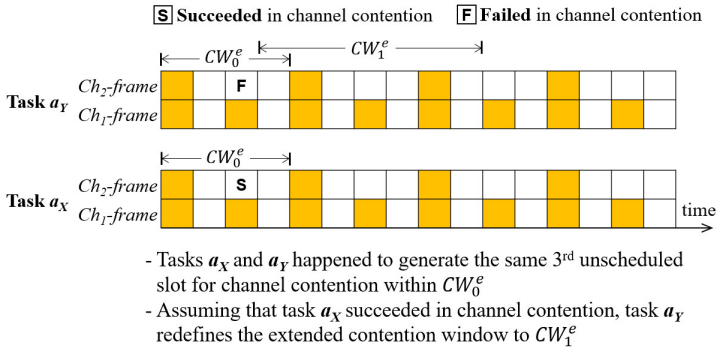


Fig. 6. Illustration of the two-level collision avoidance scheme with $|CW_0| = 4$

If task a_X generates *rdelay* smaller than task a_Y, it acquires channel 2 (*Ch2-frame*), and succeeds in sending data, while task a_Y defers its transmission and executes the L1-CA scheme again. If the two tasks generate the same *rdelay* during L2-CA, they will transmit data concurrently. In this case, GW can receive only one of the data correctly due to the capture effect.

E. Slot Utilization

Let us compare the *slot utilization* (SU) of periodic tasks for the two approaches ZFD and SFD. In this comparison, we consider only one group, since different groups take the same scheduling process. To enable real-time slot scheduling for every periodic task, frame period T_{frm} using ZFD is determined as the shortest of the transmission periods for periodic tasks, as follows:

$$T_{frm}(ZFD) = \min\{p_i | \ \tau_i \in G\} \tag{12}$$

where $p_i = \alpha_i \times T_{frm}(ZFD)$, $\alpha_i \in N^+$. Given n tasks in G, $SU(ZFD)$ with ZFD can be calculated as follows:

$$SU(ZFD) = {}^1\!/_{\mathrm{n}} \sum_{j=1..n} {}^1\!/_{\alpha_j} \tag{13}$$

Just for convenience in the analysis, we assume that a frame period with SFD is the longest of the transmission periods for tasks, as follows:

$$T_{frm}(SFD) = \max\{p_i | \ \tau_i \in \mathrm{G}\} \tag{14}$$

Then, each task acquires the exact number of required slots within the frame period by using the LSI algorithm. Thus, no transmission slot is wasted, regardless of the differences in transmission periods. Thus, since SFD does not waste slots, $SU(SFD)$ is given *unity*:

$$SU(SFD) = 1 \tag{15}$$

Let us take an example to compare channel utilization under the two approaches. Consider two task groups, as follows:

$$G_1 = (\tau_A, \tau_B, \tau_C) = \big((\mathrm{A}, 8), (\mathrm{B}, 8), (\mathrm{C}, 8)\big)$$

and

$$G_2 = (\tau_X, \tau_Y, \tau_Z) = \big((\mathrm{X}, 8), (\mathrm{Y}, 16), (\mathrm{Z}, 32)\big)$$

The frame period and slot utilization of each approach are calculated in Table III.

TABLE III
FRAME PERIOD AND SLOT UTILIZATION OF ZFD AND SFD FOR TWO TASK GROUPS.

	$T_{frm}(ZFD)$	$T_{frm}(SFD)$	$SU(ZFD)$	$SU(SFD)$
G_1	8	8	1	1
G_2	8	32	0.58	1

Since the transmission periods of all tasks in G_1 are eight slots long, both ZFD and SFD set T_{frm} to 8. Thus, both approaches can fully utilize the slots. With G_2, ZFD will set T_{frm} to 8 while SFD sets it to 32. Thus, with ZFD, task τ_Y and task τ_Z use one slot every two and four frame periods, respectively, wasting one slot and three slots, respectively. Meanwhile, SFD can still achieve maximum slot utilization.

IV. PERFORMANCE EVALUATION

Since the transmission of periodic data relies on slot scheduling, we focus on evaluating the performance of aperiodic data transmission. The proposed protocol, RTLoRa-LFP using the SFD approach, was compared with ILoRa [28] and RT-LoRa [29] using the ZFD approach. The compared protocols were implemented as follows. The ILoRa protocol used the Pure Aloha approach for transmission of aperiodic data. The RT-LoRa protocol employed Slotted Aloha where, if an aperiodic task arrives within the CFP, it

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2022.3162019, IEEE Internet of Things Journal

randomly selects a transmission slot within the forthcoming CAP to avoid a collision, and transmits data in the selected slot; however, if the aperiodic task instead arrives within the CAP, it selects the next slot to transmit data due to the nature of random arrival.

A. Simulation Model

For performance evaluation, a simulation tool was developed based on the well-known discrete event simulator, Simpy [32]. Since Simpy does not provide LoRa simulation model, we borrowed LoRa communication model that defines communication range and collision behavior from LoRaSim [17]. In LoRaSim, the communication range for both gateway and end node is modeled using the following log-distance path loss model:

$$L_{pl}(d) = \overline{L_{pl}}(d_0) + 10\gamma log\left(\frac{d}{d_0}\right) + X_\sigma \quad (16)$$

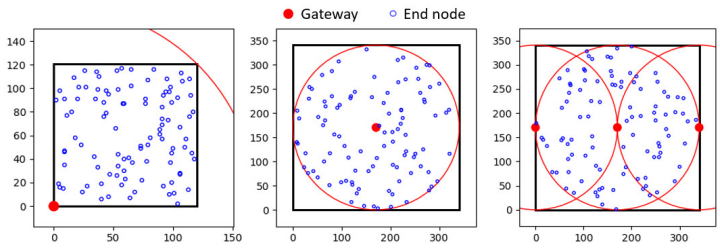
where d is the distance from node to gateway, $L_{pl}(d)$ is the path loss in decibels, $\overline{L_{pl}}(d_0)$ is the mean path loss at reference distance d_0, γ is the path loss exponent, and X_σ takes the normal distribution with zero mean and σ^2 variance to account for shadowing, represented as $X_\sigma \sim N(0, \sigma^2)$. Empirical measurements on a real node with $d_0 = 40$ give the parameter values in (16): $\overline{L_{pl}}(d_0) = 127.41\ dB$, $\gamma = 2.08$, and $\sigma = 3.57$. In this model, σ is assumed to be zero for simplicity. If a node transmits data at distance d from the GW, the GW calculates the reception power based on transmission power and $L_{pl}(d)$ from (16). Then, GW will receive data successfully if its reception power is greater than the sensitivity threshold of the receiver.

We also borrowed the collision behavior model of concurrently transmitted data x and y, $C(x, y)$, as follows:

$$C(x,y) = O(x,y) \wedge C_{freq}(x,y) \wedge C_{sf}(x,y) \wedge C_{pwr}(x,y) \wedge C_{cs}(x,y) \quad (17)$$

where $O(x,y)$ is true if reception intervals overlap by any amount; $C_{freq}(x,y)$ is true if the overlapping of frequencies is larger than a threshold value; $C_{sf}(x,y)$ is true if they use the same spreading factor; $C_{pwr}(x,y)$ is true if the difference in reception power is larger than the threshold value, and $C_{cs}(x,y)$ is true if the overlapping of reception intervals is larger than the specified value. This model implies that packets x and y always collide if all terms on the right side of (17) are true; otherwise, one or two of them can be received depending on the conditions of each term.

B. Simulation Setup



(a) S1 (b) S2 (c) S3

Fig. 7. The three deployment scenarios for the performance evaluation

One key factor that degrades the reliability of transmission in LoRa networks is the high probability of collision. To resolve this problem, the proposed approach employs a collision avoidance scheme using the LBT mechanism. However, the LBT mechanism has a limitation in dealing with the hidden node problem. Another factor to consider is *node density*, as the number of nodes (*nNodes*) divided by the size of the dimension, that determines data traffic. In industrial safety monitoring and control applications, nodes are often distributed densely within a limited area. The use of multiple GWs may alleviate the packet loss problem under heavy traffic.

As depicted in Fig. 7, three scenarios (S1, S2, and S3) were used to examine how well the compared LoRa protocols can deal with node density and the hidden node problem.

In S1, a GW is located at the corner, and end nodes are randomly distributed in a square area such that one GW covers all end nodes, where the red curved line indicates the transmission range of the GW. This scenario eliminates the hidden node problem completely that occurs when two nodes transmit a packet to the GW but are not within mutual transmission range. In S2, with the bigger dimension, the GW is located at the center, and end nodes are distributed randomly in a circle such that the GW covers all the nodes. In S3, two GWs are additionally placed at diametrically opposite points on the circumference of a circle to reduce the number of nodes exposed to the hidden node problem.

To better examine the capabilities of different MAC protocols in dealing with aperiodic tasks, we introduce one parameter to show a scheduled slot index, *ssi*, which is used to indicate the proportion of scheduled slots to total slots, as follows:

$$ssi = \frac{nSch_i}{2^N} \leq 1 \quad (18)$$

Since a higher *ssi* value implies fewer unscheduled slots (and thus, intensified channel contention among the aperiodic tasks), it can better reveal a good scheme in handling collisions and the hidden node problem. Other parameters and values are summarized in Table IV.

TABLE IV
SIMULATION PARAMETERS AND VALUES.

Parameter	Value	Parameter	Value
nNodes	50 ~ 600	*N*	8
m	1	UL slot size	100 ms
SF	7	DL segment	200 ms
Code rate	4/5	Frame period	25800 ms
Bandwidth	125 kHz	k_i (SF7)	2
Tx Power	14 dBm	($\|CW_0\|$, *MaxCW*)	(4, 64)
Packet size	35 bytes	*MaxDelayCnt*	10
λ	1/(25800 ms)	CE threshold	3 dBm

According to Table IV, *m* was given *unity* and SF7 only was used in the simulation. The reason is that it was sufficient to use only one channel in distinguishing the performance of different protocols, and the LoRa network with SF7 can cover the work field of around 350 × 350 m^2 and support multiple hundreds of end nodes. Note that the larger areas can be covered by the use of multiple GWs. Each UL slot of 100 ms can accommodate the packet of 50 bytes which have been used typically for industrial safety monitoring and control applications [25, 29]. For aperiodic data, each UL slot can accommodate the transmission time of 35 bytes (< 79.52 ms) with SF7, thereby reserving the remaining 20.48 ms for maximum *rdelay* ($MaxDelayCnt \times delayslot = 20.48\ ms$ from Table II). Note that the large aperiodic data is segmented before transmission. It is

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2022.3162019, IEEE Internet of Things Journal

assumed that the arrival of aperiodic data at each node follows the Poisson process at the rate of $\lambda = 1/(25800\ ms)$, considering the following scenario. A sensor node may have to produce aperiodic data more frequently if it enters areas with restricted access or detects dangerous signs like fire. Sometimes, a node with multiple sensor modules may have to report aperiodic data including the status change (power on or off) of an individual sensor module to save energy less frequently. Even though this scenario happens during some time periods, the protocol should be able to handle such data traffic properly. Finally, some protocols make use of the capture effect (CE) in data transmission, thereby requiring the CE threshold value that is used to judge the occurrence of collision. In this study, the CE threshold was set to 3 dBm according to [30], whereas LoRaSim used 6 dBm.

Some evaluation metrics are defined as follows. One of them is *packet delivery ratio* (PDR) defined as the ratio of data packets received at the server to those transmitted by all nodes. The second one is *average delay* defined as the elapsed time from when a node generates a packet to the time the server receives the packet. The delay includes a binary backoff delay, a random delay within a slot, the CAD time, and the packet ToA; it does not include the delay in the backhaul network. The third one is *fairness* of data delivery, defined as the degree to which every node can have an equal chance of success in packet transmission without being sacrificed by the capture effect. Fairness is examined by comparing the PDRs of nodes in terms of the median, the maximum, the minimum, the first quartile, and the third quartile values.

C. Results and Discussion

1) Effect of the Contention Window

Simulation was performed with S1 to investigate the sensitivity of RTLoRa-LFP when varying *nNodes* from 50 to 600 and varying the maximum allowable number of channel contentions (*MaxCCs*) from 1 to 7. This scenario prevents the hidden node problem. The *ssi* was set to *zero*, implying that periodic tasks do not exist.

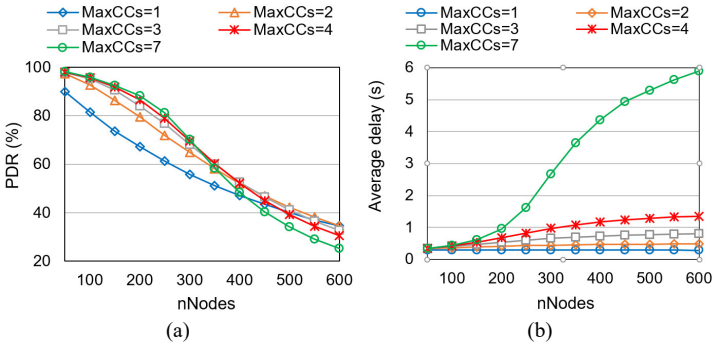


Fig. 8. Evaluation of (a) PDR and (b) average delay under RTLoRa-LFP when varying *nNodes* and *MaxCCs* (*S1*, *ssi* = 0)

Referring to Fig. 8-(a) and Fig. 8-(b), as *nNodes* goes over 250, the larger *MaxCCs* makes both PDR and delay drastically worse. This comes from the principle "the more reattempts at channel contention, the more the competing nodes". Contrarily, the protocol with *nNodes* at 256 or less showed a higher PDR for the larger *MaxCCs*, because it gives more chances to a node that failed in channel contention. If *nNodes* is 100 and *MaxCCs* is 3, the protocol can achieve a PDR over 95% while limiting the delay to below 300 ms. Considering PDR and delay, the optimal value of *MaxCCs* was chosen as 3 or 4. *MaxCCs* was set to 4 for the rest of the discussion.

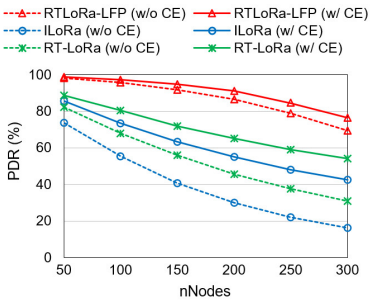


Fig. 9. PDR under different protocols with the capture effect enabled or disabled (S1, *ssi* = 0)

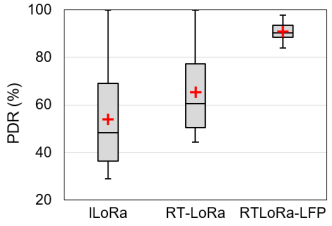


Fig. 10. The PDR distribution of nodes under different protocols (*nNodes* = 200, S1, *ssi* = 0)

Fig. 9 shows which protocol can better avoid collisions among the three protocols (RTLoRa-LFP, ILoRa, and RT-LoRa). Each protocol shows two graphs corresponding to two modes, one with the capture effect enabled (w/ CE) and another with the capture effect disabled (w/o CE). Recall that if two or more nodes transmit data concurrently, a GW demodulates only the data with the strongest signal by the capture effect. RTLoRa-LFP showed a small gap in the PDRs between two modes, while the other protocols showed a big gap. This implies that RTLoRa-LFP removed the cases of concurrent transmissions considerably by using the *two-level collision avoidance* scheme. It is worth noting that the other approaches rely on the capture effect considerably.

Fig. 10 shows that the PDR distribution of nodes varied from 29% to 100% and from 44% to 100% under ILoRa and RT-LoRa, respectively, whereas it varied from 84% to 98% under RTLoRa-LFP. It is observed that RTLoRa-LFP achieves high fairness in data transmission. Fairness can also be quantified by applying the following Jain's fairness equation [33]:

$$f(x_1, x_2, \ldots, x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} {x_i}^2} \tag{19}$$

where n and x_i indicate the number of aperiodic tasks and the PDR of an aperiodic task i, respectively. From (19), the fairness indices for ILoRa, RT-LoRa, and RTLoRa-LFP are given 0.88, 0.94, and 1.0, respectively.

2) Effect of Logical Frame Partitioning

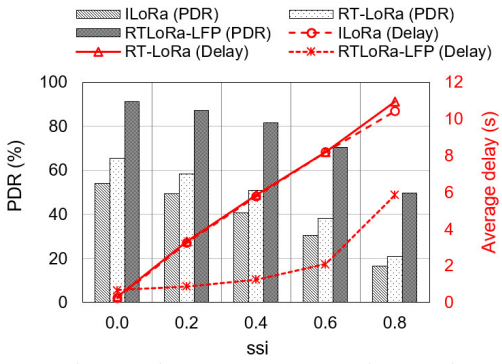


Fig. 11. PDR and average delay according to increases in *ssi* (*nNodes* = 200, S1)

Simulation was performed to evaluate the effect of the frame partitioning scheme by using different *ssi* values. As shown in Fig. 11, all three protocols showed decreasing PDRs from increasing the *ssi* value, because the reduced number of unscheduled slots increases the degree of channel contention. However, RTLoRa-LFP outperformed the other approaches clearly due to the use of the two-level collision avoidance scheme.

As for the transmission delay, when *ssi* = 0.0, both ILoRa and RT-LoRa showed a delay slightly lower than RTLoRa-LFP because they can immediately transmit aperiodic data upon arrival. As the size of the CFP increased with the increase in *ssi*, aperiodic data that arrived during the CFP could not be transmitted before the start of the CAP, thereby increasing the delay. However, under the RTLoRa-LFP protocol, since unscheduled slots are distributed over the frame, the waiting times were much shorter than under the other two protocols, on average. Note that the RTLoRa-LFP curve for average delay stayed almost flat till *ssi* increased to 0.6. For the high *ssi* value of *0.9*, we summarize the PDR and delay in Table V due to the scaling problem with the large values.

TABLE V
PDR AND AVERAGE DELAY AT SSI = 0.9.

	ILoRa	RT-LoRa	RTLoRa-LFP
PDR	8.1	10.2	31.7
Delay	8.0	10.5	43.0

If *ssi* increases to the extremely large value of 0.9, the number of unscheduled slots is only 25 in a frame of 256 slots in this simulation. The average delay under RTLoRa-LFP increased sharply to 43 seconds while delay under the other protocols decreased slightly. This is because RTLoRa-LFP tries to transmit data over multiple frame periods, thereby achieving a relatively high PDR and a long delay; however, the others achieved low PDRs because most of the packets were lost early due to collisions.

3) Effect of Multiple GWs

Simulation of the three protocols was performed to evaluate the effect of using multiple GWs in different scenarios (S1, S2, and S3). Note that a node succeeds in data transmission if any of the GWs receives the data in LoRa networks. Therefore, with the use of more GWs, nodes better overcome two interference types, one by neighboring nodes and another by hidden nodes.

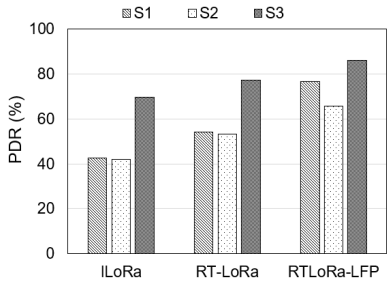


Fig. 12. PDR under different scenarios (*nNodes* = 200)

First, let us compare ILoRa and RT-LoRa in S1 and S2. ILoRa and RT-LoRa employ Pure Aloha and Slotted Aloha, respectively, that do not distinguish between the interference types. Thus, referring to Fig. 12, it is obvious that RT-LoRa achieved a PDR higher than ILoRa, while each of them achieved almost the same PDR for S1 and S2. Second, as for RTLoRa-LFP, the PDR in S2 was lower than in S1 by almost 10%. This is because many nodes in S2 were exposed to the hidden node problem. Also observed is that RTLoRa-LFP achieved a much higher PDR than the others. This implies that the two-level collision avoidance scheme can handle the collision problem. On the other hand, all three protocols improved the PDR in S3 considerably, compared to the other two scenarios. For ILoRa and RT-LoRa, the improvement comes from the use of two additional GWs and the capture effect, whereas for RTLoRa-LFP, the reason of improvement is that many nodes are insulated from the hidden node problem owing to the two additional GWs, and the data transmissions of those nodes were treated with the two-level collision avoidance scheme.

V. CONCLUSION

The RTLoRa-LFP protocol not only guarantees that periodic tasks finish their data transmissions within the specified periods, but it also allows aperiodic tasks to transmit their data with high reliability and a short delay. The former property is achieved by using slot scheduling, while the latter improvement is achieved by employing the two-level collision avoidance scheme using a contention window and a delay slot. Simulation results with various scenarios indicated that RTLoRa-LFP can achieve a high packet delivery ratio with a short delay in transmitting aperiodic data. This approach can be considered for the standardization process of real-time IoT networks because it provides a real-time data transmission service in LoRa networks.

REFERENCES

[1] M. Bor, J. Vidler, and U. Roedig, "LoRa for the Internet of Things," *EWSN,* pp. 361-366, 02/15 2016.

[2] J. P. S. Sundaram, W. Du, and Z. Zhao, "A Survey on LoRa Networking: Research Problems, Current Solutions, and Open Issues," *IEEE Communications Surveys & Tutorials,* vol. 22, no. 1, pp. 371-388, 2020.

[3] J. Haxhibeqiri, A. Karaagac, F. V. d. Abeele, W. Joseph, I. Moerman, and J. Hoebeke, "LoRa indoor coverage and performance in an industrial environment: Case study," in *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2017, pp. 1-8.

[4] A. Malik, J. Qadir, B. Ahmad, K.-L. Alvin Yau, and U. Ullah, "QoS in IEEE 802.11-based wireless networks: A contemporary review," *Journal of Network and Computer Applications,* vol. 55, pp. 24-46, 2015/09/01/ 2015.

[5] E. Ferro and F. Potorti, "Bluetooth and Wi-Fi wireless protocols: a survey and a comparison," *IEEE Wireless Communications,* vol. 12, no. 1, pp. 12-26, 2005.

[6] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks,* vol. 38, no. 4, pp. 393-422, 2002/03/15/ 2002.

[7] R. S. Sinha, Y. Wei, and S.-H. Hwang, "A survey on LPWA technology: LoRa and NB-IoT," *ICT Express,* vol. 3, no. 1, pp. 14-21, 2017/03/01/ 2017.

[8] P. Baronti, P. Pillai, V. W. C. Chook, S. Chessa, A. Gotta, and Y. F. Hu, "Wireless sensor networks: A survey on the state of the art and the 802.15.4 and ZigBee standards," *Computer Communications,* vol. 30, no. 7, pp. 1655-1695, 2007/05/26/ 2007.

[9] G. Anastasi, M. Conti, and M. D. Francesco, "A Comprehensive Analysis of the MAC Unreliability Problem in IEEE 802.15.4 Wireless Sensor Networks," *IEEE Transactions on Industrial Informatics,* vol. 7, no. 1, pp. 52-65, 2011.

[10] R. Daidone, G. Dini, and G. Anastasi, "On evaluating the performance impact of the IEEE 802.15.4 security sub-layer," *Computer Communications,* vol. 47, pp. 65-76, 2014/07/01/ 2014.

[11] D. De Guglielmo, S. Brienza, and G. Anastasi, "IEEE 802.15.4e: A survey," *Computer Communications,* vol. 88, pp. 1-24, 2016/08/15/ 2016.

[12] V. C. Gungor and G. P. Hancke, "Industrial Wireless Sensor Networks: Challenges, Design Principles, and Technical Approaches," *IEEE Transactions on Industrial Electronics,* vol. 56, no. 10, pp. 4258-4265, 2009.

[13] B. Reynders, W. Meert, and S. Pollin, "Range and coexistence analysis of long range unlicensed communication," in *2016 23rd International Conference on Telecommunications (ICT)*, 2016, pp. 1-6.

[14] A. Augustin, J. Yi, T. Clausen, and M. W. Townsley, "A Study of LoRa: Long Range & Low Power Networks for the Internet of Things," *Sensors,* vol. 16, no. 9, 2016.

[15] 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, 2017.

[16] J. d. C. Silva, J. J. P. C. Rodrigues, A. M. Alberti, P. Solic, and A. L. L. Aquino, "LoRaWAN — A low power WAN protocol for Internet of Things: A review and opportunities," in *2017 2nd International Multidisciplinary Conference on Computer and Energy Science (SpliTech)*, 2017, pp. 1-6.

[17] M. C. Bor, U. Roedig, T. Voigt, and J. M. Alonso, "Do LoRa Low-Power Wide-Area Networks Scale?," presented at the Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, Malta, Malta, 2016.

[18] O. Georgiou and U. Raza, "Low Power Wide Area Network Analysis: Can LoRa Scale?," *IEEE Wireless Communications Letters,* vol. 6, no. 2, pp. 162-165, 2017.

[19] K. Mikhaylov, P. Juha, and T. Haenninen, "Analysis of Capacity and Scalability of the LoRa Low Power Wide Area Network Technology," in *European Wireless 2016; 22th European Wireless Conference*, 2016, pp. 1-6.

[20] F. V. d. Abeele, J. Haxhibeqiri, I. Moerman, and J. Hoebeke, "Scalability Analysis of Large-Scale LoRaWAN Networks in ns-3," *IEEE Internet of Things Journal,* vol. 4, no. 6, pp. 2186-2198, 2017.

[21] T. Polonelli, D. Brunelli, A. Marzocchi, and L. Benini, "Slotted ALOHA on LoRaWAN-Design, Analysis, and Deployment," (in eng), *Sensors (Basel, Switzerland),* vol. 19, no. 4, p. 838, 2019.

[22] Q. L. Hoang, H. P. Tran, W.-S. Jung, S. H. Hoang, and H. Oh, "A Slotted Transmission with Collision Avoidance for LoRa Networks," *Procedia Computer Science,* vol. 177, pp. 94-101, 2020/01/01/ 2020.

[23] C. Pham, "Robust CSMA for long-range LoRa transmissions with image sensing devices," in *2018 Wireless Days (WD)*, 2018, pp. 116-122.

[24] T. To and A. Duda, "Simulation of LoRa in NS-3: Improving LoRa Performance with CSMA," in *2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1-7.

[25] M. Rizzi, P. Ferrari, A. Flammini, E. Sisinni, and M. Gidlund, "Using LoRa for industrial wireless networks," in *2017 IEEE 13th International Workshop on Factory Communication Systems (WFCS)*, 2017, pp. 1-4.

[26] D. Zorbas and B. O. Flynn, "Autonomous Collision-Free Scheduling for LoRa-Based Industrial Internet of Things," in *2019 IEEE 20th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, 2019, pp. 1-5.

[27] Q. L. Hoang, W. 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.

[28] 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 *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*, 2018, pp. 4141-4146.

[29] L. Leonardi, F. Battaglia, and L. L. Bello, "RT-LoRa: A Medium Access Strategy to Support Real-Time Flows Over LoRa-Based Networks for Industrial IoT Applications," *IEEE Internet of Things Journal,* vol. 6, no. 6, pp. 10812-10823, 2019.

[30] D. Croce, M. Gucciardo, S. Mangione, G. Santaromita, and I. Tinnirello, "Impact of LoRa Imperfect Orthogonality: Analysis of Link-Level Performance," *IEEE Communications Letters,* vol. 22, no. 4, pp. 796-799, 2018.

[31] Semtech, "Application Note: SX126x CAD Performance Evaluation," November 2019 2019.

[32] N. J. D. Matloff, CA. Dept of Computer Science. University of California at Davis. Retrieved on August, "Introduction to discrete-event simulation and the simpy language," vol. 2, no. 2009, pp. 1-33, 2008.

[33] R. K. Jain, D.-M. W. Chiu, and W. R. J. E. R. L. Hawe, Digital Equipment Corporation, Hudson, MA, "A quantitative measure of fairness and discrimination," 1984.