LoRaHART: Hardware-Aware Real-Time Scheduling for LoRa

Soumya Ranjan Sahoo
Nanyang Technological University, Singapore

Amalinda Gamage
National University of Singapore, Singapore

Niraj Kumar
Indian Institute of Technology, Goa, India

Arvind Easwaran
Nanyang Technological University, Singapore

Abstract

Time-sensitive data acquisition is critical for many Low-Power Wide-Area Network (LPWAN) applications, such as healthcare monitoring and industrial Internet of Things. Among the available LPWAN technologies, LoRa (Long Range) has emerged as a leading choice, offering kilometer-scale communication with minimal power consumption and enabling high-density deployments across large areas. However, the conventional ALOHA-based Medium Access Control (MAC) in LoRa is not designed to support real-time communication over large-scale networks. This paper introduces LoRaHART, a novel approach that overcomes two critical, under-explored limitations in Commercial Off The Shelf (COTS) LoRa gateways that impact real-time performance. LoRa gateways have limited capacity for demodulation of parallel transmissions and their antenna can either transmit or receive at any time instant. LoRaHART incorporates a hardware-aware super-frame structure, comprising both Time Division Multiple Access (TDMA) slots as well as opportunistic retransmissions using Carrier Sense Multiple Access (CSMA), designed to mitigate the above constraints. We use a partial packing and makespan minimization algorithm to schedule periodic real-time transmissions efficiently within the TDMA slots, and also develop a probabilistic node contention model for CSMA retransmissions, providing analytical guarantees for deadline satisfaction under ideal channel conditions. Our evaluation of LoRaHART on a 40-node LoRa testbed demonstrates significant improvements over existing solutions in practice, achieving an average Packet Reception Ratio of 98% and a 45% higher airtime utilization than the best performing baseline.

2012 ACM Subject Classification Networks → Network protocols; Networks → Sensor networks; Computer systems organization → Embedded and cyber-physical systems; Software and its engineering → Real-time systems software

Keywords and phrases LoRa, LPWAN, Real-time Scheduling, Hardware Constraints

Digital Object Identifier 10.4230/LIPIcs.ECRTS.2025.17

Funding This work was supported by the MoE Tier-2 grant MOET2EP20221-0006.

1 Introduction

The demand for real-time data collection and actuation in IoT networks is rapidly expanding, with applications spanning from residential to industrial automation. With projections suggesting up to 50 billion IoT devices by 2030 [1], the shared Industrial, Scientific and Medical (ISM) spectrum is expected to become increasingly crowded, as more devices compete for the same bandwidth. Efficient use of this spectrum is therefore essential to address this growing interference, which will pose significant challenges for time-sensitive applications. In addition, many IoT devices are energy-constrained, and expected to operate for up to 10 years on a single battery [2]. These devices often operate hundreds of meters or even

kilometers away from base stations. Therefore, robust, long-range communication protocols are key to maintain efficient data transmissions. To meet these goals, Low-Power Wide-Area Networks (LPWANs) serve as the enabling technology [3].

Real-time LPWANs are indeed desirable across multiple domains. In smart cities, they can enable real-time traffic monitoring and control to prevent congestion and enhance road safety [4]. In healthcare, they can facilitate remote patient monitoring through timely communication of critical health data [5]. In industrial automation, they can support machine monitoring and predictive maintenance [6]. In environment monitoring, they can enable early detection of forest fires and volcanic activity in remote regions, thereby enhancing disaster prevention [7]. By delivering kilometers wide communication ranges at milliwatt power budgets, LPWAN technologies are indispensable for such real-time IoT applications [2, 8].

Several technologies, including LoRa, SigFox, and NB-IoT, are specifically designed for LPWANs. LoRa, in particular, stands out for its high scalability and versatility. A single LoRa gateway can handle up to 10,000 nodes [9, 10], and achieves extensive coverage, with line-of-sight communication distances exceeding six kilometers in some deployments [2, 8, 11]. LoRa's unique implementation of Chirp Spread Spectrum (CSS) modulation provides both extended range and high multiple-access capacity. The long-range capability results from the inherent properties of CSS modulation, while LoRa's adaptation of CSS allows transmissions with different Spreading Factors (SFs) to coexist within the same channel—a feature known as *orthogonality*. Thus, LoRa's orthogonality combined with its operation on the unlicensed ISM band, enables cost-effective, large-scale deployments.

While LoRa's default ALOHA-based MAC offers simplicity, it is best suited for sporadic, non time-sensitive applications. Although prior works propose CSMA adaptations for LoRa, these solutions are not optimized for real-time, latency-sensitive applications [12–14]. This limits LoRa's ability to meet the bounded latency requirements necessary for time-critical tasks. To address these challenges, recent studies have explored MAC designs for LoRa that aim to achieve real-time performance [15–17]. Real-Time LoRaWAN Scheduler (`RTLS`), which enhances the default ALOHA-based MAC layer protocol in LoRa (called LoRaWAN) with a network-layer earliest deadline first scheduler was recently proposed [15]. However, adding real-time capabilities to LoRaWAN presents inherent challenges, such as the mandatory Rx (reception) window required after each uplink message [18]. The Rx window consumes considerable airtime, which decreases the time available for real-time scheduling. Additionally, `RTPL` (Real-Time Communication Protocol for LoRa Networks), which utilizes a TDMA-based protocol to replace the ALOHA-based MAC layer has also been proposed [16, 17]. However, the assumptions made in this work regarding the capability of LoRa concurrent transmissions and the duplex nature of COTS LoRa gateways severely limit the protocol's applicability and reliability. In this work, we address the aforementioned shortcomings and propose a novel MAC layer real-time protocol for LoRa called `LoRaHART`, making the following fundamental contributions:

1. We conduct experiments on an industry-grade LoRa testbed to showcase two main limitations in COTS LoRa hardware applicable for real-time applications. We conclude that it is essential to respect the *uplink-downlink asymmetry* and *concurrency limits* innate to COTS LoRa hardware – Section 3.
2. We propose a hardware-aware super-frame structure for `LoRaHART` to enable real-time periodic transmissions using COTS hardware. In addition to the TDMA-based real-time uplink messages to the gateway, the super-frame also supports transmission of acknowledgments from the gateway and provides a CSMA-based opportunistic retransmission capability to handle transmission failures – Section 5.

3. We develop a probabilistic model for node contention in the opportunistic retransmission window (CSMA) of `LoRaHART`. This model provides a closed-form expression for the probability of a successful transmission under given contention parameters, enabling us to derive probabilistic real-time guarantees under ideal channel conditions – Section 5.1.
4. We tackle the problem of real-time channel and super-frame scheduling in `LoRaHART` (TDMA slots) by addressing three sub-problems: packing messages into super-frames, assigning messages to channels within each super-frame, and computing a schedule for messages assigned to each channel. We develop an iterative scheduling technique based on partial packing and makespan minimization with prioritization using the Rate Monotonic (RM) strategy – Section 6.
5. We extensively evaluate `LoRaHART` on a 40-node industry-grade LoRa testbed. We compare our approach with two baselines, `RTPL` [16] and `RTLS` [15] achieving an average improvement of 20% in PRR and 45% in airtime utilization against the best performing baseline. We also conduct experiments to validate the resilience of `LoRaHART` in the presence of faults by injecting spurious transmissions – Section 7.

Related Works. While real-time protocols like RT-Wi-fi [19] and WirelessHART [20] use centralized TDMA-based scheduling to ensure bounded latency in real-time settings, they do not have LoRa's unique capabilities such as SF-based orthogonality which allows multiple nodes to transmit concurrently on the same channel without interfering.

In LoRa, only a few recent studies have investigated its performance for real-time applications [7, 15–17, 21].

As mentioned earlier, `RTLS` [15] replaces the default ALOHA protocol with a real-time scheduler that computes schedules for a multi-gateway LoRa system. In this design the nodes are first grouped based on their given (fixed) SFs and assigned random channels. Then a non-preemptive schedule is generated using the Earliest Deadline First (EDF) strategy for each group. At most 6 concurrent transmissions are possible in this design due to SF-based groups. This, together with the mandatory LoRaWAN Rx window, implies that the spectral efficiency of the protocol is very low. Furthermore, absence of retransmission mechanisms to handle failures makes this design less reliable.

The authors in [16] proposed a MAC layer protocol, namely `RTPL`, which first maps nodes to partitions defined by a unique combination of channel and SF. The scheduler then allocates uplink/downlink transmissions across these partitions using EDF. However, `RTPL` overlooks certain critical aspects of LoRa hardware, such as the inability of a LoRa gateway to reliably demodulate more than 8 packets simultaneously, by allowing up to 48 (6×8) concurrent transmissions. Additionally, `RTPL` schedules uplink and downlink transmissions concurrently, disregarding the half-duplex nature of LoRa gateways, which can cause it to miss incoming uplink transmissions while being occupied with downlinks. `RTPL` also suggests using multiple radios to send concurrent downlinks to several nodes at once, which is impractical for a COTS LoRa gateway. Furthermore, `RTPL` prioritizes the nodes based on decreasing order of SFs (high to low) for mapping to partitions, which can penalize nodes transmitting at a lower SF. In contrast, our scheduling approach prioritizes based on the rate monotonic policy and uses only the minimum required SF for each node, while respecting the gateway's ability of demodulating up to 8 concurrent transmissions.

Burst-MAC [7] addresses sudden burst traffic in LoRa by temporarily adopting a semi-distributed TDMA approach under high-load conditions. Specifically, Burst-MAC confines the collision domain of each node within small groups, permitting concurrent transmissions from multiple such groups. It employs a hash-based method for slot assignment within each group, eliminating the overhead of centralized schedule distribution typically associated with

Table 1 Comparison of LoRa-based Real-Time MAC Protocols

Feature	LoRaHART	RT-LoRa	RTPL	RTLS	Burst-MAC
Underlying MAC	TDMA + CSMA	TDMA + ALOHA	TDMA	TDMA	TDMA (semi-distributed)
Retransmission	Yes (LMAC)	No	Yes (Slot-based)	No	Yes (Hash-based)
COTS Limitations Respected	Yes (up to 8)	No (up to 48)	No (up to 48)	Yes (up to 6)	No (up to 384)
Scheduling Approach	RM + makespan heuristic	all transmissions in each super-frame	EDF with partitioning	EDF	Hash-based TDMA
Communication Mode	Uplink	Uplink/Downlink	Uplink/Downlink	Uplink	Uplink
Gateway Duplex Mode	Half-duplex	Half-duplex	Full-duplex	Half-duplex	Full-duplex

traditional TDMA schemes. However, despite these advantages, Burst-MAC is not designed for regular (periodic) real-time transmissions, and its allowance of up to 384 (64 channels x 6 SFs) parallel transmissions inherently violates LoRa gateway hardware constraints thereby raising concerns for real-world deployment scenarios.

The authors in [21] propose a MAC strategy called `RT-LoRa`, designed for real-time transmissions in industrial IoT applications. `RT-LoRa` organizes network time into super-frames with dedicated, yet flexible, durations for uplink and downlink transmissions. `RT-LoRa` accommodates both stationary and mobile nodes and offers three Quality of Service (QoS) classes, balancing transmission reliability against energy overhead. However, `RT-LoRa` relies on an impractical assumption that all the periodic real-time transmissions share the same generation period and as a consequence are scheduled in every super-frame. Thus, they do not address the channel and super-frame scheduling problem for more general periodic real-time transmissions considered in this paper; if the transmission periods are not all the same, then a deadline-aware mapping of transmissions to super-frames is necessary for feasible scheduling. In the later sections of this paper we provide a detailed comparison between `LoRaHART` and `RT-LoRa` (Section 4 compares the super-frame structures, Section 6.3 briefly discusses the differences in scheduling requirements, and Section 7.2 presents the resulting implications on performance).

We summarize the key characteristics of each of the above protocols including `LoRaHART`, highlighting their differences and suitability across deployment scenarios in Table 1.

In 802.11-based Wi-Fi, extensive modeling efforts have been made to analyze the collision probabilities under both saturated (nodes always have backlogged packets) and unsaturated (with a packet arrival rate) conditions [22–24]. Bianchi's seminal paper [25] provides a two-dimensional Markov chain model, which derives success and collision probabilities along with overall throughput for CSMA/CA (Collision Avoidance). However, these models do not directly apply to LoRa's CSMA protocol, where nodes are typically unsaturated and lack acknowledgments preventing feedback-driven retransmissions. Our work bridges this gap by developing a probabilistic model tailored to LoRa's CSMA contention, providing an analytical framework for transmission success probabilities.

2 A Primer on LoRa

In this section, we present a brief background on the LoRa physical layer, highlighting its distinctive advantages and the role of LoRa Channel Activity Detection in enhancing communication efficiency [26].

A typical LoRa network consists of many nodes distributed over a large geographical area, all communicating with one or more gateways. An *uplink* (UL) refers to a communication from a node to a gateway, whereas a *downlink* (DL) is a communication from a gateway to a node. Downlinks are required to deliver acknowledgments (ACKs) to nodes.

LoRa communication takes place over the ISM band and the specific band a network utilizes is a *regional parameter*; for instance, in the US, LoRa uses the 900 MHz ISM band.

LoRa segments the available ISM band into multiple *channels* (more specifically, frequency channels), each capable of supporting both *uplink* and *downlink* communications.

LoRa and Orthogonality. The strength of LoRa lies in its use of Chirp Spread Spectrum (CSS) Modulation. CSS provides LoRa a competitive edge over other LPWAN technologies by facilitating longer communication distances and higher multiple access for networks. LoRa's physical layer supports simultaneous non-disruptive transmissions on the same channel through the use of different Spreading Factors (SFs) known as *orthogonality* [27–30]. Orthogonality plays a key role in enhancing multiple access in LoRa networks and differentiates it from the rest of LPWAN modulations. Orthogonality of LoRa significantly enhances network coexistence and multiple access by allowing concurrent transmissions within the same channel. LoRa supports six SFs, ranging from SF7 to SF12. A lower SF allows smaller airtime and lower energy consumption for the same payload, but has a reduced transmission range. However, the use of orthogonality is limited by hardware capabilities and, therefore, should be a factor accounted for when designing scalable real-time LoRa (details in Section 3).

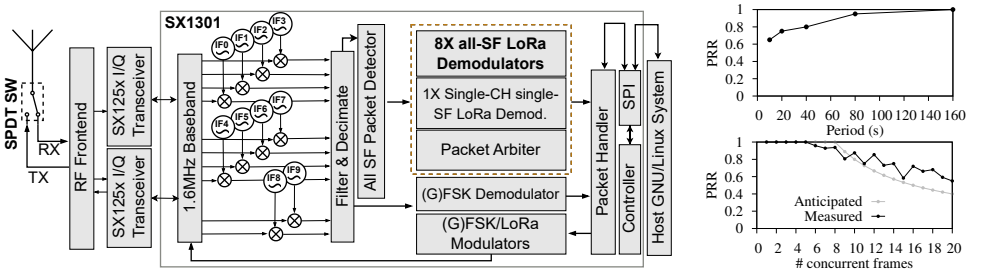
Collisions. A LoRa collision occurs when two or more nodes transmit concurrently in the same channel with the same SF [31–33]. Therefore, whenever a LoRa packet is transmitted asynchronously (without an agreed upon schedule), using CSMA prior to transmission is desirable to minimize collisions. However, LoRa modulation is unique in that LoRa packets can be demodulated beneath the noise floor [8, 34, 35] and allows concurrent transmissions within a single channel. Due to this unique nature, a CSMA mechanism that doesn't cripple the said features is required [36–38]. CSMA designs from other wireless technologies, e.g., WiFi [39, 40], utilize the received signal strength measurements to verify the presence of ongoing transmissions. However, LoRa networks will yield inefficiency from such signal strength based protection as it cripples LoRa's orthogonality and misses below noise packets [12]. To this end, we adopt LMAC [13], which exploits the low-power Channel Activity Detection (CAD) mode available within all LoRa radios to avoid collisions. Every CAD operation is SF-selective, i.e., it can detect transmissions in a specified SF while remaining insensitive to other simultaneous transmissions in different SFs within the same channel. Thus, adopting LMAC allows preserving LoRa's multiple access capability.

3 Motivation

Understanding the limitations of a wireless technology is key to designing an efficient network utilizing the same. In this regard, we identify two key limitations that hinder LoRa's effectiveness in large-scale real-time deployments. 1) Uplink-downlink asymmetry: a COTS LoRa gateway operates in half-duplex mode, meaning it cannot receive uplink transmissions while busy with a downlink transmission. 2) Limits on concurrency: a COTS LoRa gateway can only demodulate up to 8 simultaneous uplinks [41]. This section introduces the architectural constraints of COTS LoRa gateways and provides experimental evidence to illustrate their impact[1].

Uplink-Downlink Asymmetry. LoRa uplink transmissions can fail for several reasons including external channel interference. To address this, LoRa nodes can rely on downlink messages from gateways to determine if a *retransmission* is necessary upon a failed

[1] The design of the testbed used for the experiments in this section is discussed in detail in Section 7.



(a) Gateway Architecture **(b)** Packet Reception Ratio

Figure 1 Architecture and Concurrency Limits of a COTS LoRa Gateway

transmission. Downlinks are thus essential for designing real-time systems that require high reliability. A LoRa gateway integrates functionality to cater to uplinks and downlinks, however, not both simultaneously since COTS LoRa gateways operate in half-duplex mode. Thus, a LoRa gateway loses all incoming uplinks while it is busy with a downlink transmission. We have studied the Semtech documentation on the standard LoRa gateway as well as its reference design and present the architecture of a COTS LoRa gateway in Figure 1a [41, 42]. The fundamental reason for this uplink-downlink asymmetry is that the antenna path of the LoRa gateway physically shifts between *receive* and *transmit* paths, causing the gateway to miss all uplink data during downlink transmission. More specifically, the Radio Frequency (RF) switch, present in LoRa gateways, temporarily disconnects the incoming RF data from the demodulators during transmission – a hardware constraint that cannot be bypassed with software solutions. Although some preliminary works have aimed to address this issue by introducing full-duplex capabilities to LoRa gateways [43, 44], those require modifications to COTS hardware and reduces the multiple access capability of LoRa.

Recent research, such as Busy Signal Multiple Access (BSMA) [43], has shown that it is possible to build a custom full-duplex LoRa gateway capable of transmitting a busy signal while receiving uplink packets. Their prototype achieved true full-duplex operation but revealed significant barriers to commercial adoption. To function correctly, BSMA's gateway needs to suppress its own transmitted signal by over 100 dB, matching LoRa's extreme receiver sensitivity threshold of nearly -120 dBm. Achieving this level of cancellation requires complex analog circuitry, including specially tuned vector modulators, custom approximations for short-delay multi-path effects, and substantial physical antenna isolation. Even with these modifications, careful manual adjustment of antenna placement is necessary to minimize interference. These technical demands impose high costs, increased design complexity, and larger physical footprints. As a result, deploying full-duplex gateways at scale remains commercially nonviable at the moment.

We conducted an experiment using our testbed and observed considerable loss in the Packet Reception Ratio (PRR) due to the uplink-downlink asymmetry, as illustrated in the top plot in Figure 1b. In this experiment, 16 nodes periodically communicate with a COTS LoRa gateway, and we observe the PRR for varying periods. Note that these periods also denote transmission deadlines for the nodes. Upon transmission, each node requests an ACK, which requires the gateway to switch into transmit mode. As shorter periods imply more ACKs, the gateway is forced to spend longer duration in transmission mode, being *deaf* to incoming packets. As a result, the shorter the node period, the higher the ratio of lost packets.

Limits on Concurrency. Although the communication bandwidth of a LoRa channel consumes only 125KHz, a single LoRa channel is designed to be 200kHz to accommodate

guardbands to minimize interference [45]. A gateway therefore continuously processes 1.6MHz of baseband covering 8 LoRa channels. As illustrated in Figure 1a, the two SX1257 front-end transceivers collectively digitize 1.6MHz of ISM spectrum and pass this data to the SX1301 digital baseband chip for further processing. The SX1301 detects, demodulates, and decodes this data, and finally transfers packets to the host system.

A single LoRa channel can accommodate simultaneous packets with different SFs. Therefore, the All SF Packet Detector continuously scans all 8 channels (IF0 to IF7) for potential preambles belonging to all SFs, whereas channels IF8 and IF9 are reserved for gateway-gateway communication and frequency shift-keying modulation, respectively. Once a preamble is detected, the *Packet Arbiter* assigns its demodulation to one of the available 8 demodulators. In this manner, the SX1301 baseband chip architecture separates the preamble detection process from demodulation, also shown in Figure 1a. Therefore, the gateway can simultaneously detect packets from any of the 48 possible SF/channel combinations. *However, it can only demodulate up to* 8 *packets simultaneously due to the* 8 *demodulators* [41]. Note, the additional single-channel single-SF demodulator highlighted in Figure 1a is pre-configured at boot time and designed only to be utilized as a backhaul link between gateways.

The bottom plot in Figure 1b illustrates the observations from experiments which validates the said limitation. As the number of concurrent packets transmitted towards the gateway reach 8, further increments lead to a sharp decrease in PRR, translating to missed deadlines. The anticipated line represents the theoretical PRR, assuming the gateway is limited to receiving only 8 concurrent packets at a time. Note, although Figure 1b shows occasions where more than 8 packets could sometimes be demodulated due to buffering, the gateway should not be operated under such concurrency. This is because packet loss in a wireless network is mainly due to the unreliable nature of the wireless channel. Operating beyond the rated capabilities of the SX1301 chip further amplifies this packet loss.

4 System Model

We consider a LoRa-based LPWAN system with a set of n *nodes* and a single *gateway*. While multi-gateway setups are possible in LoRa, we focus on a single gateway setup in this work; a single LoRa gateway can support up to 10000 nodes over a coverage range of up to 6km [8, 9].

Each node transmits a periodic real-time *message* to the gateway. Let $m_1, \ldots, m_n$ represent the n periodic messages from the nodes, where each message m_i has a period p_i, also its deadline. Without loss of generality, we assume the periods are sorted, that is $p_i \leq p_{i+1}$ for each i. We also assume that each p_i is an integer multiple of p_1, a typical assumption for industrial and IoT applications that often rely on harmonic periods [20]. Note, this is a weaker condition than strict harmonicity, and it enables us to define a fixed-length super-frame of duration p_1 such that all message deadlines are aligned with super-frame boundaries. The *hyper-period* is defined as $h = \text{lcm}\{p_1, \ldots, p_n\}$. Let $m_{i,j}$ denote the j^{th} instance of m_i, which is generated at time instant $(j-1) \times p_i$ and must complete transmission by time instant $j \times p_i$.

Let set $\mathcal{S} = \{\text{SF7}, \ldots, \text{SF12}\}$ denote the SFs available in LoRa. The smallest feasible SF, sf_i, for message m_i is determined by the node-gateway distance and we assume this is given. The *airtime* of a data packet, which includes header as well as cyclic redundancy check, is determined by message size, SF and modulation parameters, and it can be obtained using Eq.7 in [8].

A message can be transmitted in any one of the 8 channels $\mathcal{C} = \{ch_1, \ldots, ch_8\}$. The transmissions across different channels do not interfere even if transmitted with the same SF.

Furthermore, different SFs can be used by nodes transmitting in the same channel without interference. Thus, in theory, up to 6×8 messages can be transmitted concurrently without collisions. *However, as explained in Section 3, a COTS LoRa gateway can only demodulate up to* 8 *messages concurrently, and hence we use this practical limit in our setting.*

In a *hyper-frame* having duration equal to hyper-period, messages are scheduled in p_1-duration *super-frames* using TDMA. The structure of this super-frame is discussed in detail in Section 5. Let these super-frames be labeled as $\{F_0, F_1, \ldots, F_{h/p_1-1}\}$. Up to 8 concurrent transmissions can occur at each time instant in any super-frame, due to the aforementioned practical limit. It is then reasonable to schedule those 8 concurrent transmissions on 8 different channels to minimize inter-packet noise. Further, to minimize transmission time and energy, each instance of message m_i is always transmitted using the smallest feasible SF. Although it is possible to schedule transmissions using higher SFs, we show in Section 6 that there is no loss in schedulability by considering only the smallest feasible SFs. Consequently, the required *slot* length for message m_i, denoted L_i, which includes both the airtime (calculated with the smallest feasible SF) and any additional network overhead, can be determined.

The slot for a message instance $m_{i,j}$ must be reserved in one of the 8 channels during the activation window $((j-1)p_i, jp_i]$. We assume that each message instance must complete transmission within the same super-frame. Let $ST_{i,j}$ be the start time of transmission for $m_{i,j}$; $ST_{i,j}$ is time relative to the start of the hyper-frame. Then, this transmission will finish at $FT_{i,j} = ST_{i,j} + L_i$ when the message is received at the gateway, and we require that time instants $ST_{i,j}$ and $FT_{i,j}$ belong to the same super-frame.

Thus, given a LoRa physical layer based real-time system described above, the problems addressed by the proposed `LoRaHART` protocol are:

P1. Design a super-frame structure with provisions for deterministic transmissions using TDMA, acknowledgments from the gateway for TDMA transmissions, and opportunistic retransmissions for failures using CSMA.

P2. Model the contention dynamics of the CSMA-based retransmission phase to compute the probability of successful transmissions under ideal channel conditions.

P3. Design a channel and super-frame scheduling algorithm for meeting deadlines of messages $m_1, \ldots, m_n$ using TDMA-based transmissions under ideal channel conditions.

Due to possible external interferences to the wireless medium (non-ideal channel conditions), retransmissions may be required for failed TDMA-based uplinks. An efficient acknowledgment mechanism is then crucial so nodes can detect transmission failures. Since transmission status is unknown at design time, deterministic retransmission slots would waste network resources. Therefore, we propose a super-frame structure that supports multicast acknowledgments and provides opportunistic CSMA-based retransmission capability. However, since these retransmissions may have collisions with each other, we also derive a model to analyze such transmission collisions and obtain a probabilistic estimate of successful retransmissions. Eventually, to evaluate the practical performance of `LoRaHART` (under non-ideal channel conditions), we conduct extensive testbed experiments discussed in Section 7.2.

For TDMA scheduling of channels and super-frames, we determine the following three parameters for each message instance $m_{i,j}$ in the hyper-period: a super-frame $F_{i,j} \in \{F_x | x \in \{(j-1)p_i/p_1, \ldots, jp_i/p_1 - 1\}\}$ and channel $C_{i,j}$ to which the instance is mapped, and a start time $ST_{i,j}$ for transmission of the data packet satisfying the condition $FT_{i,j} \leq jp_i$. Additionally, $ST_{i,j}$ and $FT_{i,j}$ must belong to the same super-frame, and for collision free transmissions one must also ensure that there is no overlap in the slots allocated to message instances that are scheduled in the same channel.

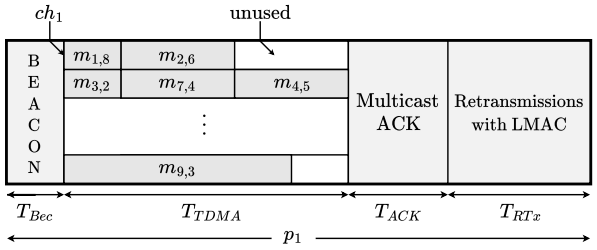


Figure 2 `LoRaHART` Super-frame Structure

5 Super-Frame Structure

In this section, we address Problem P1 and present a hardware-aware super-frame structure for enabling real-time transmissions in LoRa-based networks. The proposed structure supports real-time uplink messages via TDMA, acknowledgments and opportunistic retransmissions for failed transmissions via CSMA, all while accounting for the hardware limitations described in Section 3 – uplink-downlink asymmetry and concurrency constraints.

Transmissions occur in repeated super-frames, each with a duration of p_1. A hyper-period h consists of h/p_1 super-frames, collectively called a *hyper-frame*. Each super-frame is divided into four segments, as illustrated in Figure 2: Beacon (Bec) of duration T_{Bec}, TDMA of duration T_{TDMA}, Acknowledgment (ACK) of duration T_{ACK} and Retransmission (RTx) of duration T_{RTx}. The super-frame begins with a *Beacon* segment Bec, which upon reception enables the nodes to synchronize their time and start their uplinks to the gateway.

The second segment, TDMA, operates using a Time Division Multiple Access protocol where each message instance is assigned a *slot* for transmission. Furthermore, by leveraging channel orthogonality, up to 8 uplinks can be transmitted concurrently at each time instant in this segment. The spreading factor (SF) determines the slot length for each transmission, and slots are assigned within the activation window $((j-1)p_i, jp_i]$ for each message instance $m_{i,j}$. Importantly, transmissions must be completed within the same super-frame, as preemption is not allowed. The scheduling algorithm for determining the slot parameters $\langle F_{i,j}, C_{i,j}, ST_{i,j}\rangle$ for each message instance $m_{i,j}$ in this segment is discussed in Section 6.

Following the TDMA segment is the acknowledgment, ACK, segment. It is crucial that acknowledgment messages are sent in time to allow retransmissions when needed. A unicast ACK, which is asynchronous with uplinks from other nodes, would require the gateway to frequently switch between receive and transmit modes, causing uplink losses. To address this, we implement *multicast acknowledgment* [21], where the gateway sends a collective acknowledgment for all successfully received transmissions in each super-frame. This can be implemented by multicasting a bit vector, where the i^{th} bit indicates the success (bit-1) or failure (bit-0) of a transmission as shown in Figure 3. The multicast ACK is transmitted under the highest SF (SF12) on a predetermined channel. Although the number of bits in the bit vector depends on the number of nodes, this approach remains scalable for practical network sizes, e.g., a 3 second T_{ACK} window is sufficient to accommodate the airtime required for multicasting a bit vector comprising 600 nodes. Furthermore, our ACK method significantly reduces the gateway's transmission time, allowing it to spend more time *listening* rather than *transmitting*. This increased listening time reduces uplink losses as the gateway is no longer occupied with per-uplink ACK transmission. To the best of our knowledge, we are the first to incorporate and evaluate this aggregated ACK on a large-scale LoRa testbed.

In the case of a failed transmission from a node, the node gets an opportunity to retransmit in the same super-frame; more specifically, following the ACK segment, a segment

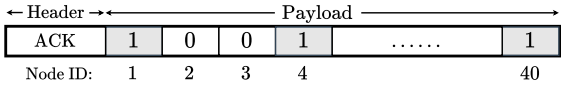


Figure 3 Multicast Acknowledgment

referred to as the RTx segment is reserved for retransmissions. Unlike the TDMA segment, a retransmission slot is not reserved a priori, but instead the nodes with failed transmissions attempt to retransmit by following LMAC (an efficient CSMA-based protocol for LoRa) [12]. To improve the chance of a successful retransmission, we propose to retransmit a message instance with a SF that is one level higher than its smallest feasible SF (or SF12 if it is already the smallest feasible SF). Note that this approach allows for a greater number of potential retransmissions in the RTx segment compared to retransmitting with SF12 for each failed node, since the slot durations are shorter. The number of concurrent transmissions within the RTx segment are not limited to 8 but managed by LMAC.

Together, these four segments form a super-frame. Recall our assumption that $p_i = y \times p_1$ (where y is a positive integer) for each message m_i. This ensures that all message deadlines align with super-frame boundaries. Moreover, synchronizing downlinks allows us to avoid the issue of uplink-downlink asymmetry. Considering the COTS LoRa gateway demodulation capability, the proposed super-frame limits the maximum number of concurrent transmissions at any time instant to 8 (as discussed in Section 3).

`LoRaHART` employs a flexible super-frame structure that can be adapted to different deployment scenarios and can be scaled efficiently with network size. In its current configuration the 5-second retransmission (RTx) window proves to be sufficiently large where very few number of nodes (4–5[2]) typically use it in a single super-frame, leaving most of the window idle. Furthermore, reducing packet sizes below 26 bytes allows shorter TDMA slots and increases the number of packets within each super-frame. Smaller packet sizes and improved synchronization accuracy at the sub-second level will enable further slot size reductions, minimizing airtime wasted on buffer intervals. Parameters such as T_{ACK}, T_{RTx}, and slot durations are therefore intentionally designed to be configurable. They should be tuned carefully based on network requirements, informed by experimental evaluation in the intended deployment environment.

Comparison of Super-frame Structures between `RT-LoRa` and `LoRaHART`: As shown in Figure 4, `RT-LoRa` organizes its super-frame into five sections: (i) a Beacon window transmitting across all Spreading Factors (SFs) for synchronization and mobility, enabling dynamic SF selection, allowing mobile nodes to choose the optimal SF from the successfully received beacons. In contrast, `LoRaHART` assumes precomputed minimum feasible SF assignments suited for energy-constrained static networks; (ii) a Contention Access Period (CAP) supporting unscheduled aperiodic uplinks via slotted ALOHA; (iii) a Contention-Free Period (CFP) for periodic real-time transmissions using multi-channel, multi-SF TDMA scheduling; (iv) an optional Downlink window for gateway-to-node communications; and (v) a CFP Acknowledgment window for global feedback corresponding to CFP uplinks, identical to the multicast acknowledgment window in `LoRaHART`. One major distinction is that, unlike `LoRaHART`, `RT-LoRa` does not utilize this feedback mechanism for retransmissions of undelivered periodic real-time messages. Later in Sections 7.2 and 7.4, we experimentally show the robustness benefits of the RTx window in `LoRaHART`.

[2] Details are presented in experimental validation of LMAC under Section 7.6

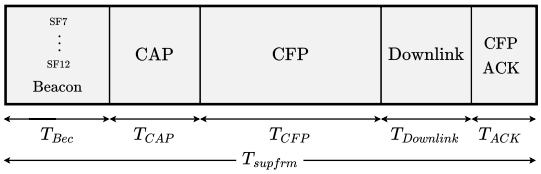


Figure 4 Super-frame structure of RT-LoRa [21]

5.1 Probability Model for Transmission Collisions in LMAC

In this section, we address Problem P2 by presenting a probabilistic model for node contention in the retransmission (RTx) window. The nodes adhere to LMAC [12] in this window and we assume ideal channel conditions, meaning that if only one node transmits within a given time window the transmission is considered successful from the sender's perspective. We focus on the contention process of a single channel and SF, where k out of n nodes — having a 0-bit in their acknowledgment vector in the ACK window — compete for access to their respective ch/sf. Since channels and SFs other than the selected ones remain orthogonal, they do not contribute to contention. However, a successful transmission from the sender does not guarantee reception at the gateway. As shown in Figure 1b, even when concurrent transmissions are limited to 8, the gateway does not achieve 100% PRR, highlighting inherent reception limitations. Practical constraints — such as external interferences and decoding inefficiencies — prevent perfect demodulation. Modeling such reception dynamics would require a more comprehensive model, which is beyond the scope of this work.

Overview of LMAC. LMAC is inspired by the 802.11 Distributed Coordination Function (DCF) [46]. When a node initiates a transmission on its assigned Channel/SF (ch/sf), it first waits for a Distributed Inter-Frame Space (DIFS) duration, during which it performs 12 Channel Activity Detection (CAD) operations. If all CADs report an idle channel, the node randomly selects a backoff value $N_{\text{BO}} \in \{0, 1, \ldots, b-1\}$ and periodically senses the channel and decrements this counter if the channel is idle. Here b is an integer value which can be chosen empirically based on the number of nodes. In the event that the channel is sensed to be busy during this countdown, the node *freezes* its backoff counter, reverts to DIFS mode where it performs CAD, and resumes the countdown only when the channel is idle again. Moreover, nodes do not receive acknowledgments for data packets transmitted during this LMAC window; thus, each node may only get a single chance to transmit its message within RTx. For more details on LMAC, please see [45].

Modeling Assumptions. We derive a closed-form expression for the probability of a successful transmission ($P_{success}$) under the assumption that the maximum backoff value b for a ch/sf is limited by the ratio T_{RTx}/T_{tx}, where T_{tx} represents the airtime of a message. This is a reasonable assumption because it ensures that in the worst-case, where backoff countdown freezes for a duration of T_{tx} at each value, when the countdown completes there is still sufficient remaining time for the node to attempt a transmission in RTx. Furthermore, since any node transmitting after the current transmission must wait for a DIFS interval before sending a packet, we merge this DIFS duration into T_{tx} for simplicity.

Probability of Successful Transmission. Suppose $N_{\text{BO}} = s \leq b-1$ for some node. Note, $s < T_{RTx}/T_{tx}$ by the above assumption and a node transmits only when the backoff counter

reaches 0. Since backoff values are chosen uniformly at random from the range $\{0, \ldots, b-1\}$, the probability of selecting any particular value s is: $P_{\text{tx}} = 1/b$. A transmission is considered successful if no other node selects the same backoff value s. The probability that none of the remaining $(k-1)$ nodes select the same value s can be given as: $P_{success} = (1 - 1/b)^{k-1}$.

Expected Number of Successful Nodes. Since each of the k nodes is successful with probability P_{success} (independently), the expected number of total successful transmissions for a ch/sf combination having k contending nodes in the retransmission window can be obtained via linearity of expectation as: $\mathbb{E}[\#\text{ successes}] = \sum_k P_{success} = k \cdot P_{\text{success}}$.

6 Super-Frame and Channel Scheduling

In this section, we address the scheduling problem P3, which involves determining the super-frame $F_{i,j}$, channel $C_{i,j}$, and slot start time $ST_{i,j}$ for each message instance $m_{i,j}$ within the hyper-period h. Once a super-frame from the set $\{F_x | x \in \{(j-1)p_i/p_1, \ldots, jp_i/p_1 - 1\}\}$ is selected, we ensure that the timing constraint $FT_{i,j} \leq (x+1)p_1 - (T_{RTx} + T_{ACK})$ is met, *i.e.*, the transmission must finish within the TDMA segment of the selected super-frame. *A set of message instances is considered feasible for transmission in a super-frame if each instance can be scheduled on one of the* 8 *channels, with non-overlapping slot allocations in each channel and with each transmission completing within the TDMA segment.*

We solve problem P3 by addressing the following three sub-problems: `SP1`) Determine $(m_{i,j} \mapsto F_{i,j})$ a packing of each instance $m_{i,j}$ to one of the p_i/p_1 super-frames within $((j-1)p_i, jp_i]$ such that all the instances packed to a super-frame are feasible. `SP2`) Determine $(m_{i,j} \mapsto C_{i,j})$ a packing of instances to channels within each super-frame. `SP3`) Compute $(m_{i,j} \mapsto ST_{i,j})$ a schedule of instances in each channel of each super-frame such that the transmission finishes within the TDMA segment and the slot allocations are non-overlapping.

To address these sub-problems, we propose: (i) Message Instance to Super-Frame Packing (`MFP`) Algorithm to solve `SP1`, and (ii) Message Instance to Channel Packing (`MCP`) Algorithm for `SP2` and `SP3`, which returns false if a set of message instances mapped to a super-frame are infeasible and schedules the instances in channels otherwise. The `MFP` algorithm prioritizes message instances based on the *rate monotonic* strategy. `MCP` is used to check the feasibility of packing a message instance into a super-frame. If the packing is found to be feasible, the message is packed and scheduled in a channel in the corresponding super-frame.

6.1 Message Instance to Channel Packing (`MCP`) Algorithm

In this section, we propose `MCP`, an algorithm to pack and schedule message instances that are mapped to a specific super-frame in 8 channels. For each instance $m_{i,j}$ in the super-frame, the objective is to determine $C_{i,j}$, the channel to which it is packed, and $ST_{i,j}$, its allocated slot start time. `MCP` has to ensure that the slot finish time $FT_{i,j} = ST_{i,j} + L_i$ is within the TDMA segment of the super-frame and that slot allocations in a channel are non-overlapping.

The `MCP` algorithm adopts the Sum Partial Solutions (SPS) algorithm from [47], which addresses the problem of mapping non-preemptive jobs to identical machines to minimize makespan. SPS algorithm works in two phases: firstly, using the Determining Partial Solutions (DPS) algorithm, a tuple of partial packings are obtained, which are then iteratively merged in the second phase to compute a final packing of the jobs to machines. We modify SPS to fit the context of channel packing and scheduling in LoRa networks, where channels are analogous to machines and message instances to the non-preemptive jobs. Furthermore, SPS

Algorithm 1 `MCP`($\mathcal{I}$) Algorithm

```
   /* Phase 1 */
1  Rearrange indices in I such that L_i ≥ L_{i+1} for all i
2  Initialize y ← 1 and N_p ← 1
3  Set P_y = ⟨P_{y,1} = {m_1}, P_{y,2} = ∅, ..., P_{y,8} = ∅⟩ and add P_y to P
4  Set Δ_y = ⟨Δ_{y,1} = L_1, Δ_{y,2} = 0, ..., Δ_{y,8} = 0⟩ and δ_y = L_1
5  for i = 2 to |I| do
6      if L_i ≤ δ_1 then
7          Add m_i to P_{1,8}
8          Update Δ_{1,8} ← Δ_{1,8} + L_i
9          Rearrange indices in P_1 in decreasing order of Δ_{1,l} and update δ_1
10     else
11         N_p ← N_p + 1; y ← N_p
12         Set P_y = ⟨m_i, ∅, ..., ∅⟩ and add P_y to P
13         Set Δ_y = ⟨L_i, 0, ..., 0⟩ and δ_y = L_i
14     Rearrange indices in P in decreasing order of δ_y
   /* Phase 2 */
15 while |P| > 1 do
       /* Merge P_1 and P_2 */
16     for l = 1 to 8 do
17         P_{1,l} ← P_{1,l} ∪ P_{2,8−l+1}
18     Rearrange indices in P_1 in decreasing order of Δ_{1,l}
19     Update Δ_1 and δ_1
20     Discard P_2 from P
21     Rearrange indices in P in decreasing order of δ_y
22 for l = 1 to 8 do
23     if Δ_{1,l} > T_TDMA then
24         return False
25 return P_1
```

attempts to minimize the makespan, whereas any makespan smaller than the duration of the TDMA segment of the super-frame is acceptable for `MCP`.

Given a tuple of message instances $\mathcal{I}$, `MCP` either returns a feasible packing and schedule for the 8 channels or reports failure. It works in two phases. In the first phase, a tuple of *partial packings* are computed such that each partial packing (i) consists of 8 *sub-groups* where each sub-group, if non-empty, consists of a subset of $\mathcal{I}$ that has been packed to the corresponding channel, and (ii) is exclusive, *i.e.*, a message instance is included in *exactly* one sub-group of a partial packing. Subsequently, partial packings are merged iteratively until only one packing remains in the second phase.

In phase 1 of `MCP` (shown in Algorithm 1), the tuple of message instances $\mathcal{I}$ is first arranged in decreasing order of slot length L_i (Line 1). The algorithm computes a tuple $\mathcal{P}$ of partial packings, where each partial packing P_y represents a subset of message instances from $\mathcal{I}$. Each partial packing consists of 8 sub-groups $P_{y,l}$ ($l \in \{1, \ldots, 8\}$), with each sub-group assigned to a specific channel. The total slot length of all instances in the l^{th} sub-group is denoted as $\Delta_{y,l}$. The *gap* of a partial packing, δ_y, is defined as the difference between the

Algorithm 2 `MFP` Algorithm

```
1  Rearrange indices in M such that p_i ≤ p_{i+1} for all i  /* RM priority */
2  foreach m_i ∈ M do
3      for j = 1 to h/p_i do
4          Initialize set of candidate super-frames cand ← {F_{(j-1)p_i/p_1}, ..., F_{(jp_i/p_1)-1}}
5          status ← False
6          foreach F_x ∈ cand considered in ascending order of x do
7              I ← set of message instances already packed in F_x
8              if MCP(I ∪ m_{i,j}) is True then
9                  status ← True; Break
10         if status = False then
11             Report Failure
```

maximum and minimum sub-group total slot lengths: $\delta_y = \max_l\{\Delta_{y,l}\} - \min_l\{\Delta_{y,l}\}$. `MCP` begins with an initial partial packing where the message instance with the largest slot length is placed in the first sub-group (lines 2 - 4). Since at most one instance of any message will be mapped to a single super-frame, we drop the subscript j in $m_{i,j}$ in this subsection. As subsequent message instances are packed, they are either added to the partial packing with the largest gap (lines 5 - 9) or a new partial packing is created (lines 10 - 13). The sub-groups within each partial packing are always kept in decreasing order of total slot lengths (Line 9), and the tuple of partial packings is maintained in decreasing order of gap size (Line 14). Let, $N_p = |\mathcal{P}|$ be the number of partial packings in $\mathcal{P}$ at the end of phase 1. Phase 1 concludes once all the message instances have been packed into a partial packing, with $\bigcup P_{y,l} = \mathcal{I}$.

In phase 2 of `MCP`, two partial packings with the highest and second highest gaps are merged. The first sub-group of the first partial packing is merged with the last sub-group of the second partial packing, and so on (lines 16 - 17). Subsequently, the sub-groups in the updated partial packing P_1 are sorted (Line 18) followed by updation of Δ_1 and δ_1. Finally, the indices in $\mathcal{P}$ are rearranged in decreasing order of gap δ_y. This procedure is repeated until $\mathcal{P}$ contains no more than one partial packing. The merged partial packing P_1 (for which $\bigcup P_{1,l} = \mathcal{I}$) is evaluated to ensure that the schedule completes within the TDMA segment of the super-frame (lines 22 - 24).

Let P^* be an optimal packing of message instances $\mathcal{I}$ produced by an *oracle* corresponding to P_1 obtained in Line 25 of Algorithm 1. We define Δ^* to denote the makespan of the optimal solution, whereas $\Delta^{mcp} = \max_{l\in\{1,\ldots,8\}}\{\Delta_{1,l}\}$ denotes the makespan of `MCP`. Then the approximation ratio for `MCP` algorithm is (this follows from the approximation ratio of SPS algorithm presented in Theorem 4.1 [47])

$$\frac{\Delta^{mcp}}{\Delta^*} \leq 1 + \frac{c-1}{c \cdot N_p} \text{ if } N_p > 1 \tag{1a}$$

$$\Delta^{mcp} = \Delta^* \text{ if } N_p = 1 \tag{1b}$$

where c is the number of channels (in this case 8), and N_p is the number of partial packings.

The initial sorting of indices in descending order of their slot length for $\mathcal{I}$ takes $O(n \log n)$ time, where n denotes the maximum number of message instances that can be packed in a super-frame. Excluding the operation on line 18 which takes $O(c \log c)$ time, note that, rearranging the indices in P_1 and P can be accomplished in logarithmic time, because only

one of the updated indices needs repositioning. Hence the creation of partial packings (loop in Line 5) takes $O(n \log n)$ time and the merging of those partial packings into one final packing (loop in Line 15) takes $O(n \log c)$ time [47], where c is number of channels. The overall computational complexity of MCP algorithm is $O(n \log n)$.

6.2 Message Instance to Super-Frame Packing (MFP) Algorithm

The MFP algorithm addresses sub-problem SP1 of packing message instances to super-frames and obtains a packing in each super-frame using MCP. It is detailed in Algorithm 2.

Messages are packed into super-frames in priority order as determined by the rate monotonic algorithm (Line 1). Messages with shorter periods have higher priority, with ties broken arbitrarily. For each message instance $m_{i,j}$, the algorithm evaluates a set of candidate super-frames within its activation window (earliest super-frame first), and checks if adding the instance is feasible using the MCP algorithm (Line 8). If no feasible packing is found, the algorithm reports failure (Line 11). Once all the message instances are packed into super-frames, the schedule for each channel and super-frame can be easily computed from the output of MCP. Recall that message deadlines are aligned with super-frame boundaries and only super-frames within the activation window of a message instance are considered for packing. Therefore, the exact ordering of messages within a channel in a super-frame does not affect the deadline guarantees. That is, given the tuple P_1 returned by MCP, each sub-group $P_{1,l}$ ($l \in \{1, \ldots, 8\}$) can be mapped to the corresponding channel l, and the message instances within $P_{1,l}$ can be scheduled in any arbitrary order.

Complexity Analysis: Messages are sorted based on periods following the rate monotonic strategy (Line 1), which takes $O(n \log n)$ time for n messages. For each of the h/p_i instances of message m_i, the algorithm evaluates up to p_i/p_1 candidate super-frames. This results in a total of $O\left(n \cdot h/p_1\right)$ invocations of MCP. Each invocation of MCP has a complexity of $O(n \log n)$, where n is the maximum number of message instances in any super-frame. Therefore, the overall complexity for MFP is: $O\left(h/p_1 \cdot n^2 \log n\right)$. Note, if periods are harmonic and the ratio of largest to smallest period can be regarded as a constant (a common setting in many real-world industrial and IoT applications [48]), this complexity becomes quadratic.

Differences in scheduling between RT-LoRa and LoRaHART: The scheduling strategy of RT-LoRa differs fundamentally from that of LoRaHART. In RT-LoRa, all periodic application layer transmissions are assumed to have the same generation period which is a very strong assumption that greatly simplifies the scheduling problem unlike the transmission model considered in this paper. Thus, each periodic real-time transmission in RT-LoRa is required to be scheduled in every super-frame, eliminating the need for a message instance to super-frame packing algorithm. Further, under RT-LoRa, nodes select an appropriate SF for their transmissions based on reception quality of the gateway beacon as well as their QoS requirements. Specifically, RT-LoRa uses SF-based grouping, where each spreading factor is mapped to a subset of available channels, and messages within the same SF group are allocated to different channels. Scheduling within a super-frame is then performed by assigning one slot per transmission with the slot length based on SF-based airtime. In contrast, LoRaHART uses a two-stage scheduling mechanism that performs both super-frame mapping and intra-frame channel packing using a packing algorithm, which provides greater flexibility for heterogeneous periods and more efficient channel utilization. Nevertheless, RT-LoRa assumes the application layer operates asynchronously with the MAC layer. As a result, a message may miss its assigned slot within the current super-frame and must

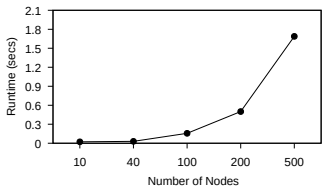


Figure 5 Runtime Complexity of Scheduling Algorithm `MFP`.

wait until the next super-frame for transmission. They provide a closed-form expression to compute this worst-case delay, and consider the generated schedule to be feasible if this delay is no more than the deadline for all the transmissions; otherwise, the transmission SF must be adjusted to generate a new schedule. Note, since the transmission model used in `RT-LoRa` is very simple (all period values are the same), their proposed scheduling approach cannot be applied for the model considered in this paper and hence we did not evaluate `RT-LoRa` in our experiments.

6.3 Scalability and SF Orthogonality of LoRaHART

To demonstrate the scalability of `LoRaHART`, we present runtime complexity measurements of the `MFP` scheduling algorithm for network sizes ranging from 40 to 500 nodes[3]. These experiments were conducted on a Linux device equipped with a 12-core Intel Xeon CPU and 32 GB RAM. Figure 5 illustrates how the runtime complexity of the algorithm (in seconds) scales as the network size increases. We observe that for relatively small networks (e.g., 40 nodes), the algorithm computes schedules efficiently, requiring only about 30 ms. For larger networks, computation time increases, reaching approximately 1.8 seconds for 500 nodes. Notably, even at this scale, the computation time remains practical and does not become a bottleneck, highlighting `LoRaHART`'s suitability for large-scale IoT deployments.

Recall from Section 4 that each message m_i is scheduled using its smallest feasible SF sf_i. In the following lemma, we show that this assumption does not impact schedulability.

▶ **Lemma 1.** *Under `LoRaHART`, if it is feasible to schedule a set of n messages, then there exists a feasible schedule in which each instance $m_{i,j}$ uses its smallest feasible SF sf_i.*

Proof. Consider any feasible schedule such that there is at least one message instance $m_{i,j}$ which is scheduled using a SF higher than its smallest feasible SF sf_i. Note the slot length used by $m_{i,j}$ is equal to or higher than L_i, because the slot length required to transmit the message using a higher SF is at least as much as the one required for sf_i. Then, we can swap the slot for $m_{i,j}$ with a slot of length L_i so that the message instance can use its smallest feasible SF for transmission. The resulting schedule is also a feasible schedule, because in `LoRaHART` no concurrent transmission will be allocated to the same channel as $m_{i,j}$. ◀

`LoRaHART` currently maps eight concurrent transmissions directly onto the eight available channels. Even if it does not fully exploit the orthogonality of LoRa's spreading factors, it still aligns with the current hardware capabilities, by delivering a practical, hardware-aware solution for today's deployments. If the hardware evolves and gateways begin to support more parallel demodulators, new opportunities will emerge. In particular, future versions of `LoRaHART` could leverage SF orthogonality to allow multiple transmissions on the same channel, provided they use different SFs. For instance, if a gateway supports 16 parallel

[3] The workload generation process for these experiments is described in detail in Section 7.2.

demodulators, the MFP algorithm could be extended to schedule two sets of transmissions per super-frame, ensuring that two overlapping transmissions on the same channel do not share the same SF. To incorporate this into the existing design, minor modifications would be required in the MCP algorithm. Specifically, during the packing phase (lines 5–14), when adding a message instance m_i to a channel l, an additional check should be introduced to verify that no other message in that channel uses the same SF within the slot under consideration. This can be enforced by augmenting each channel group $P_{1,l}$ with the set of SFs currently assigned in that slot and rejecting the addition of any new message with a duplicate SF. Similar care must be taken during the merging phase (lines 16–19), where the SF uniqueness constraint must be respected to avoid interference. This adaptation would improve channel utilization while preserving real-time guarantees, making it a promising direction for scaling LoRaHART alongside advancements in the LoRa gateway technology.

LoRaHART adaptation for dynamic networks: LoRaHART uses an offline scheduling approach, where the computed schedule is uploaded to the nodes prior to deployment. However, in real-world settings, dynamic changes such as node joins, departures or variations in required SFs (due to mobility or environmental factors) may necessitate schedule updates. Typically, nodes report such changes to the network server via metadata embedded in control messages or periodic link-quality monitoring. Once the network server detects a topology change, it can initiate re-scheduling. To avoid recomputing the entire schedule from scratch, we can use an incremental scheduling approach in LoRaHART. The existing transmission-to-super-frame mapping can be reused for all nodes whose transmission parameters (e.g., SF) remain unchanged. The scheduling algorithm can then be re-invoked with this partial mapping as the starting point. New or modified nodes can be scheduled using LoRaHART and if all new mappings are feasible, the updated schedule can be produced without impacting the rest of the network. To support this incremental scheduling, the MCP algorithm must also be adapted to ensure that existing message-to-channel assignments are preserved during re-packing. Specifically, when invoking MCP during incremental scheduling, the already scheduled messages (whose SFs or deadlines remain unchanged) should be treated as fixed assignments within their original super-frame and channel. The candidate message instances (from newly joined or modified nodes) are then packed into the remaining available channels and slots without altering the current placements. Thus, by anchoring pre-existing allocations and only attempting to fit the new/modified messages around them, the updated MCP ensures incremental packing is feasible and efficient, provided sufficient residual capacity is available.

7 Experimental Evaluation

In this section we discuss the results of extensive indoor and outdoor experiments which evaluate the performance of LoRaHART and compare it to two baselines (RTLS [15] and RTPL [16]) on our 40-node LoRa testbed. It is divided into five subsections: 1) Section 7.1, details testbed configuration and experiment workflow; 2) Section 7.2 evaluates the system's performance across different demand levels and under the three real-time scheduling algorithms with their respective super-frame structures; 3) Section 7.3 compares the schedulability and resource efficiency of the three algorithms by using them over the super-frame structure from Section 5; 4) Section 7.4 evaluates the robustness of LoRaHART in the presence of increasing interference; 5) Section 7.5 analyses energy consumption of LoRaHART; 6) Section 7.6 validates our LMAC probability model by comparing theoretical expectation with experimental data.

7.1 Design of Testbed and Experimental Workflow

LoRa testbed. We constructed a 40-node LoRa testbed to evaluate the performance of real-time scheduling algorithms on COTS LoRa hardware. Four design goals guided this setup: first, the testbed needed to support sparse deployments across a large university laboratory; second, a central controller was required to manage the firmware of all the nodes; third, each node needed a backhaul link for centralized metadata collection; and finally, synchronization across all 40 nodes was essential to enable real-time scheduling.

Backhaul links in testbeds are essential for loading firmware onto nodes and collecting ground-truth data, such as the packet count from each node, needed for performance calculations. However, LoRa's narrowband modulation limits its capacity to transfer such data. Physically connecting all 40 nodes to a single controller would simplify firmware loading and data extraction, but this would compromise the distributed nature of the testbed. Instead, we deploy ten distributed controllers, each managing four LoRa nodes, along with a central controller that collates data. These controllers, implemented as single-board computers, communicate over WiFi. The distributed controllers load firmware and relay experimental information as needed between the LoRa nodes and the central controller.

The testbed uses Heltec Stick Lite V3 [49] LoRa nodes with integrated SX1262 LoRa radios. Communication between the LoRa nodes and distributed controllers (e.g., for firmware uploads and backhaul data transfers) is facilitated via an emulated USB-serial interface. Fig. 6a illustrates the connectivity between each component of the testbed, Fig. 6b shows an image of the testbed hardware and Fig. 6c shows the outdoor gateway. To address the requirement of supporting geographically distributed nodes, distributed controllers communicate through a network tunnel [50] with the central controller. As the LoRa gateway, we utilize a COTS LoRa gateway [42, 51]. To capture realistic performance, we conduct experiments both indoors and outdoors, as these represent typical deployment scenarios for LoRa networks. Indoor settings often involve higher multipath propagation and increased interference due to walls and objects, whereas outdoor conditions are more affected by non-line-of-sight (NLoS) propagation, with no direct line-of-sight to nodes. For outdoor experiments, the same central and distributed controllers were used, but with the gateway located outside the laboratory. Unlike the indoor setup, where many nodes had line-of-sight to the gateway, the outdoor setup had no nodes in direct line-of-sight to the gateway.

The experiment workflow. Before each experiment, the central controller generates a unique pre-compiled binary program for each LoRa node, as each node's schedule is distinct, even under the same protocol. Once devices are confirmed as ready, the central controller issues a start command to initiate the experiment. Each LoRa packet in our experiments contains a 26-byte payload with the following data: 1) node ID, 2) a packet counter, 3) transmission start time, and 4) padding bits.

An experiment includes the following steps: loading a unique schedule consisting of multiple test cases into each LoRa node, nodes awaiting a start signal in terms of a gateway beacon and upon which executing super-frame by super-frame until all the test cases are completed. Upon receiving the first gateway beacon, all nodes set time $t = 0$ and transmissions follow the schedule. The schedule for multiple test cases are loaded one after the other sequentially for convenience;

The LoRa devices in our testbed utilize built-in Temperature Compensated Oscillators (TCXOs) to maintain synchronization over prolonged periods. By sampling the micro-controller time every 30 seconds over one hour, we observed no drift at the second-level accuracy. Hence, we set our slot size as a multiple of unit-second in our experiments.

7.2 Test Case Generation and System Performance Evaluation

Test case generation. In our experiments, multiple test cases were deployed in three groups consisting of 8, 24 and 40 node testbeds to cater for low, moderate and high density deployments. We fix the minimum period, p_1, to 20 seconds. This is a reasonable sensing/actuation period for real-time LPWAN applications [52]. For each test case, we selected at least four different period values — including the mandatory period of 20 seconds — and randomly chose the remaining period values such that their Least Common Multiple (LCM) did not exceed 720 seconds, due to storage limitations of the micro-controller. Each node was then randomly assigned a period from the selected list, ensuring that every period was allocated to at least one node. Finally, nodes were assigned SFs at random.

We set the following slot parameters calculated with respect to a packet size of 26 bytes. We selected the slot lengths (L) as 1 second for SF7, SF8 and SF9 transmissions, 2 seconds for SF10 and SF11 transmissions, and 4 seconds for SF12 transmissions. These are sufficient to accommodate the transmissions for the assigned SFs.

The duration allocated for the beacon segment is 2 seconds as the packet size for beacon is comparatively small and therefore requires only 2 seconds to transmit even with SF12. In our experiments, we set T_{TDMA} to be fixed at 10 seconds, followed by a multicast ACK segment of 3 seconds and a retransmission RTx segment of 5 seconds at the end of the super-frame. This ensures that even a SF12 transmission with a slot length of 4 seconds can attempt a retransmission in the RTx segment if required with an additional time of one second allocated for LMAC's DIFS and backoff mechanism.

Demand for each test case, D, is calculated as the ratio of total slot lengths occupied by the messages generated by $\mathcal{M}$ to 8 times the total time duration for an hyper-frame. Test cases were then categorized by demand levels: $D \in [0.01, 0.15]$ as low, $D \in (0.15, 0.3]$ as moderate (mod), and $D \in (0.3, 0.5]$ as high. Note that demand values higher than 0.5 are not feasible since T_{TDMA} is exactly half the duration of a super-frame in our experiments.

To achieve specific target demands ranging from 0 to 0.5, we systematically adjusted the periods and SFs of multiple nodes. Specifically, to increase demand, we decreased the periods (resulting in more frequent transmissions) or increased the SFs (leading to longer slot durations) and vice versa. These adjustments are performed iteratively on multiple nodes ensuring that the period and lcm requirements are met. Finally, the demand is recalculated after each iteration to check whether it is within an acceptable margin of ± 0.01.

Baselines. We compare with two baselines, `RTPL` and `RTLS`. We deploy `RTPL` on our testbed exactly as recommended in [16]. `RTPL` by design allocates a significantly higher duration than necessary for all SFs. For example, an SF7 packet of 26 bytes with an airtime of 61.7 ms is allocated 7 seconds as recommended in `RTPL`. In Section 7.3, we remove this limitation when

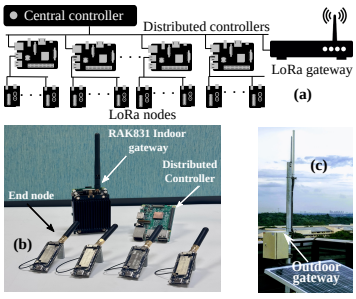
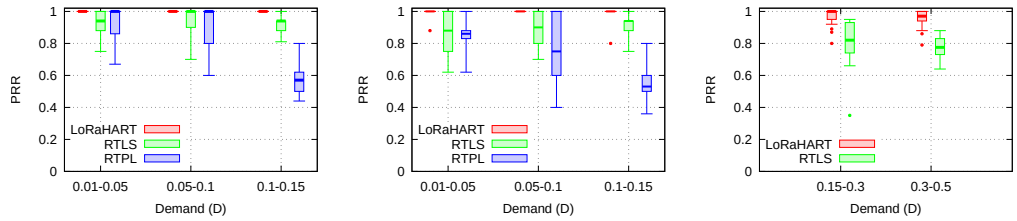


Figure 6 (a) Testbed architecture. (b) Indoor testbed nodes. (c) Outdoor gateway.



(a) Indoor (low demand) **(b)** Outdoor (low demand) **(c)** Indoor (mod/high demand)

Figure 7 Performance Comparison of Three Protocols (`LoRaHART`, `RTPL`, `RTLS`) For Indoor as well as Outdoor Experiments.

evaluating the schedulability of `RTPL` using our super-frame structure for a fair comparison. This under-utilization leads to inefficient use of the spectrum and hence `RTPL` could not schedule any moderate/high demand test cases in our experiments. For `RTLS`, we use slot lengths ranging from 3 seconds for SF7 to 15 seconds for SF12. These durations are calculated based on a 26-byte data packet as suggested in [15].

Finally, for each demand range, we selected the test cases that were deemed schedulable by the protocols (all three for the low range, and `RTLS` and `LoRaHART` for the moderate and high ranges). As such, we deployed 20 test cases for each demand range on the testbed.

Results. In Fig. 7, we present the experiment results using standard box-plots comparing the performance of the three protocols in terms of *Packet Reception Ratio* (PRR) over the hyper-period. If a packet is successfully received and demodulated by the gateway before its deadline, then it contributes to PRR. Note that a packet could fail to be delivered to the gateway due to inherent transient failures of the wireless medium. Although protocols like `LoRaHART` and `RTPL` reduce the impact of such failures by supporting retransmissions, nevertheless they cannot deterministically guarantee packet delivery. Hence, the PRR for a test case can be lower than 1 even if a test case is deemed to be schedulable by a protocol. Figs. 7a and 7b show the PRR for various values of demand in the low range ($D \in [0.01, 0.15]$) comparing the performance of all three protocols; Fig. 7a (likewise Fig. 7b) illustrates the results for indoor (likewise outdoor) experiments. Similarly, Fig. 7c illustrates the results for moderate and high demand ranges, comparing the performance of `LoRaHART` and `RTLS`.

Figs. 7a and 7b show that for the low demand range, `LoRaHART` significantly outperforms both `RTPL` and `RTLS` in terms of achieved PRR, with its mean PRR value being arbitrarily close to 1. In particular, for the indoor setting, the mean PRR value for `LoRaHART` is 0.98, which is 25% higher than the mean PRR for `RTPL` and 8% higher than the mean PRR for `RTLS`. Similarly, for the outdoor setting, the mean PRR value for `LoRaHART` is 1, which is 30% higher than the mean PRR for `RTPL` and 10% higher than the mean PRR for `RTLS`. It is noteworthy that the performance of `LoRaHART` does not vary much between the indoor and outdoor settings, illustrating its robustness and superior ability to handle transmission failures. In contrast, `RTPL` exhibits several missed deadlines, mainly due to demodulation issues arising at the gateway from more than 8 concurrent transmissions. The degradation for `RTPL` is more pronounced with increasing demand and noise (outdoor vs. indoor), which is expected. `RTLS`, on the other hand, shows better performance than `RTPL`, because by design it limits the number of concurrent transmissions to no more than 6, one per SF. However, `RTLS` still has a much lower PRR than `LoRaHART`, due to the absence of retransmissions.

In Fig. 7c we compare `LoRaHART` and `RTLS` for moderate and high demand values in the indoor setting (results are similar for the outdoor setting, but omitted due to space constraints). Here as well, `LoRaHART` significantly outperforms `RTLS`, achieving a mean PRR

of 0.97, which is 24% higher than the mean PRR for RTLS. A final remark is that both RTPL and RTLS show much greater variability in their PRR values when compared to LoRaHART, and this is because, unlike LoRaHART, they both allow multiple concurrent transmissions with different SFs in the same channel. Although LoRa specification allows for such concurrency, it typically reduces the signal-to-noise ratio leading to lower PRRs.

Recall from Section 3 that a COTS LoRa gateway has only 8 concurrent demodulators. Fig. 8 shows, from t_0 to t_1 (likewise from t_2 to t_3), a capture of all the 8 channels during the runtime of RTPL (likewise LoRaHART). It can be seen that channels cumulatively occupy more than 8 concurrent transmissions in RTPL, which cannot be successfully demodulated.

Our experimental observations reveal important performance implications for RT-LoRa when compared to LoRaHART. Like RTPL, RT-LoRa schedules the maximum number of parallel transmissions allowed, without restricting the number of CH/SF combinations. As a result, it faces the same gateway limitation on concurrent demodulations as RTPL. Under high network loads, when concurrent transmissions exceed hardware capacity, this bottleneck can lead to degraded Packet Reception Rates (PRR) as shown in Figure 7. In addition, recall that RT-LoRa does not incorporate any retransmission mechanisms for failed transmissions. This will further reduce the effective PRR for RT-LoRa even in comparison to RTPL.

7.3 Schedulability Evaluation

In this section, we evaluate the schedulability of LoRaHART, RTPL and RTLS, using their acceptance ratios which measures the fraction of test cases deemed schedulable. To ensure a fair comparison, experiments were conducted within the constraints of a COTS LoRa gateway, which limits the total number of concurrent transmissions to 8. Further, both RTPL and RTLS were adapted to use our super-frame structure along with the multicast ACK so that the underlying TDMA windows available for scheduling are identical (10 seconds in each super-frame having a total duration of 20 seconds). Additionally, we disabled the retransmission segment, RTx, since the focus of these evaluations are on schedulability.

RTLS was implemented using the same algorithm described in Section 1, with at most 6 concurrent transmissions, one per SF, and random channel assignment. Unlike RTLS, RTPL originally allows up to 48 concurrent transmissions, which exceeds the gateway's hardware capabilities. To restrict concurrency to at most 8, we limit the number of SF-channel combinations to at most 8 at any time instant, *i.e.*, we restrict RTPL to only schedule at most 8 of the 48 available SF-channel combinations at any time instant. With this restriction, we use RTPL's scheduling algorithm to allocate slots within the TDMA segments of the super-frames. Again to ensure fair comparison, we also do not schedule retransmissions under RTPL; only one transmission per message instance is scheduled within the TDMA segments. Furthermore, we utilize the same slot lengths (L) for all the protocols, i.e., 1 second for SF7, SF8 and SF9, 2 seconds for SF10 and SF11, and 4 seconds for SF12 transmissions.

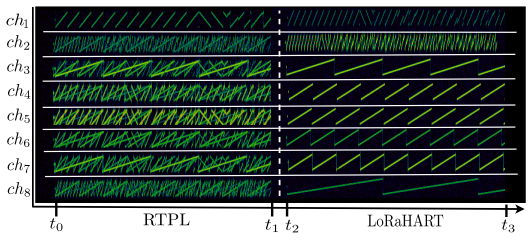
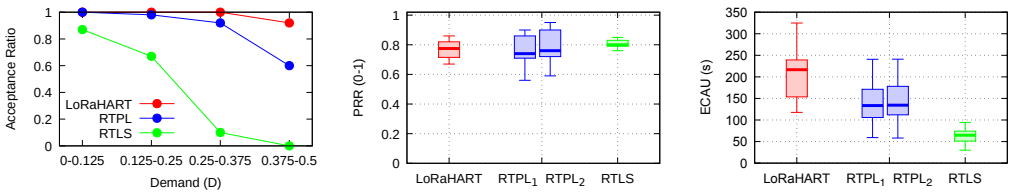


Figure 8 A Spectrogram Showing Channel Utilization



(a) Acceptance Ratio (1000 test cases) **(b)** Packet Reception Ratio (20 test cases) **(c)** Airtime Utilization (20 test cases)

Figure 9 Schedulability Comparison – `LoRaHART` vs. `RTPL` vs. `RTLS`.

Results. Fig. 9 presents the results of the schedulability experiments that were conducted. Fig. 9a shows the acceptance ratios of the three protocols when they were evaluated with 1000 test cases across a variety of demand ranges. These test cases were generated using the same technique as described in Section 7.2, with 250 test cases for each demand range shown in the figure. The acceptance ratio of `LoRaHART` is significantly higher than that of `RTLS`, clearly demonstrating the superiority of the proposed scheduling strategy. `RTLS` heavily under-utilizes available channels due to its SF based scheduling, with spectrum not being used efficiently in each frame. This accumulates significantly across the hyper-frame, resulting in reduced schedulability for high-demands. When compared to `RTPL`, `LoRaHART` has an overall acceptance ratio that is 10% higher, with the gap increasing for higher demand values. Taken together with the outcomes in Section 7.2, we can conclude that `LoRaHART` dominates both `RTPL` and `RTLS` in terms of acceptance ratio as well as PRR, and hence in terms of the ability to support LoRa-based real-time communication.

We also conducted experiments to evaluate the performance of the schedules in the above test cases that all use the same super-frame structure. For this comparison, we chose 20 schedules with the highest demands for each protocol. Note these test cases may be different for each protocol. The PRR drop in Figure 9b for `LoRaHART` validates the gain achieved by the retransmissions using LMAC shown in Figure 7. We expect PRR to be similar across the three protocols, because all the schedules respect the COTS LoRa gateway restrictions and do not allow retransmissions. Hence, we need another metric such as throughput which measures the number of packets successfully delivered for comparison. However, an algorithm that prioritizes packing packets of lower SFs may achieve higher throughput due to their shorter airtimes. Such algorithms fail to fit the diverse range of SFs necessary to accommodate nodes operating across a LoRa network. This phenomena renders throughput an inadequate performance metric in this comparison. To address this limitation, we employ a new metric – Effective Cumulative Airtime Utilization (ECAU). ECAU reflects an algorithm's packing efficiency in terms of received packets weighted by their corresponding airtime utilization. ECAU is defined as: $\sum_{m_z} T_{tx_z}$, where m_z denotes a message, T_{tx_z} denotes its airtime and the sum is computed across all the messages received within the hyper-period.

Fig. 9c shows the ECAU values for the three protocols across the 20 test cases. In the figure, $\texttt{RTPL}_1$ is the implementation of `RTPL` as described earlier in this section, and $\texttt{RTPL}_2$ is a variant which ensures that the 8 concurrent transmissions are mapped to 8 different channels similar to `LoRaHART`. On an average, the ECAU value for `LoRaHART` is 45% higher than $\texttt{RTPL}_1$ and $\texttt{RTPL}_2$ and 200% higher than `RTLS`. This clearly demonstrates that `LoRaHART` is significantly better in managing airtime at higher demands, when compared to the baselines. This, taken together with the acceptance ratios, shows that `LoRaHART` outperforms both `RTPL` and `RTLS` in terms of effectively utilizing the LoRa wireless medium for real-time communication, with the performance gap widening as message demand increases.

7.4 Robustness of LoRaHART to Fault Injections

In this section, we conduct an experiment to show the value of introducing LMAC integrated retransmission slots in the LoRaHART super-frame structure. To this end, we evaluate both RTPL and LoRaHART under three fault injection ratios: 10%, 20% and 30% of SF10 interference. An interference ratio of 10%, for instance, corresponds to a 10% demand across all 8 channels in SF10, generated by random transmissions confined strictly to the TDMA segments. To implement these conditions, we deploy 5 interferer nodes configured to transmit on SF10, collectively achieving the desired interference ratio. We repeat the experiment for each algorithm under all interference ratios. The importance of retransmission slots is seen from Fig. 10a. The retransmission slots offer gains for both the algorithms, however, due to integration of LMAC, LoRaHART uses the retransmission slots more optimally. When more than 8 nodes fail to receive acknowledgments at some instant, by design, RTPL will initiate retransmissions in the next slot for all those nodes simultaneously. This results in the transmission of large number of frames at once leading to sub-optimal reception at gateway. In contrast, the LMAC-integrated retransmission slot in LoRaHART mitigates this issue. These gains arising from efficient retransmissions also highlight the importance of effective utilization of acknowledgements under LoRaHART when compared to RT-LoRa which does not provide any such retransmission mechanism.

7.5 Energy Consumption Analysis

In the following, to quantify the energy usage for LoRaHART, we evaluate the transmission energy of a node per super-frame operating at **14 dBm**. Energy consumption can be computed as:

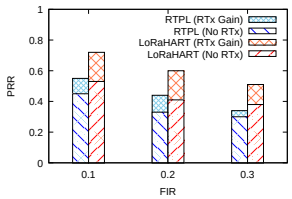
$$E_{\text{total}} = E_{\text{idle}} + E_{\text{tx}} + (1 - \text{PRR}) \cdot E_{\text{RTx}};$$

$$E_{\text{RTx}} = N_{BO} \cdot E_{\text{CAD}} + E_{\text{tx}}$$

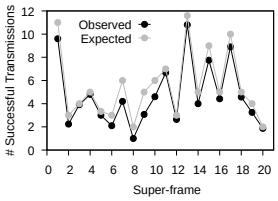
Here, E_{tx} represents the energy for a successful transmission, E_{idle} captures idle listening and beacon/ACK reception phases, and E_{RTx} accounts for retransmission energy cost when packets are lost (i.e., when PRR < 1). N_{BO} denotes the backoff duration (in slots) selected during the LMAC-based retransmission window, and E_{CAD} is the energy consumed per CAD operation. These equations reflect the specific flow and behavior of LoRaHART (including opportunistic retransmissions).

We conducted these measurements on our 40-node testbed described in Section 7.1. Energy consumption was computed by measuring the duration and current draw of each radio state—transmit, receive (idle), and CAD, using the manufacturer's specifications for the SX1301 chip [41]. Each node transmits exactly one packet per super-frame as per the TDMA schedule, and may opportunistically retransmit once in the RTx window, based on the ACK feedback.

The reported values represent average energy consumed by a node per super-frame, given by the spreading factor (SF) used. For example, SF7 nodes consume approximately 8 mJ per super-frame. In contrast, energy consumption for SF8, SF9, SF10, SF11, and SF12 nodes are observed to be approximately 20 mJ, 49 mJ, 95 mJ, 148 mJ, and 220 mJ respectively. These numbers include the energy spent in scheduled transmission, potential retransmissions, and beacon/ACK listening durations. Since a node uses a fixed SF across frames unless reconfigured, its per-frame energy consumption remains stable over time.



(a) Variation of PRR Under Different FIRs



(b) Number of Successful Transmissions in RTx

Figure 10 `LoRaHART`'s Fault Injection Tolerance and LMAC Probability Model Validation

7.6 Experimental Validation of the LMAC Probability Model

To validate our probabilistic model for node contention during the RTx window, we compare the number of successful transmissions in experiments with the theoretical expected value ($\mathbb{E}$[# successes]), given the nodes that failed in the corresponding TDMA window.

The experiment result is obtained by averaging the number of successful transmissions over 20 super-frames in the RTx window. Each failed node from the TDMA window retransmits in the RTx window using the same CH and one higher SF (except SF12 in which case we use the same SF value) when compared to the original allocation by `LoRaHART`. As previously discussed, each CH/SF pair operates independently as they are orthogonal, and hence $P_{success}$ and $\mathbb{E}$[# successes] for each CH/SF pair are calculated separately. Once the expectations for each pair are determined, summing them across all possible CH/SF groups provides the overall theoretical expectation for each super-frame.

As illustrated in Figure 10b, the theoretical predictions closely match the experimental data, with an experimental average of 4.67 and a theoretical average of 5.5 successful transmissions, exhibiting only a small deviation. Further, we observed that the theoretical expectation for each super-frame is no smaller than the corresponding experiment result, thus validating the accuracy of our probabilistic model. Under ideal channel conditions, the predicted and observed number of successful transmissions would be identical. However, external interferences and the gateway's imperfect reception introduce minor discrepancies, accounting for the gap in our idealized assumptions.

8 Conclusion

In this work, we tackled the challenge of real-time scheduling of periodic transmissions in COTS LoRa-based systems by addressing two critical limitations of standard LoRa gateways—uplink-downlink asymmetry and limited concurrency. To accommodate these constraints, we proposed a new protocol called `LoRaHART` and developed a hardware-aware super-frame structure supporting multicast acknowledgments and opportunistic retransmissions. We also devised efficient algorithms to pack messages into super-frames, assign them to channels, and schedule their transmissions. Our comprehensive evaluation on a 40-node LoRa testbed demonstrated the effectiveness of our approach in improving real-time performance and reliability. Looking ahead, we plan to consider multi-gateway setups and evaluate compliance with regulatory constraints such as the 1% duty cycle requirement. Additionally, we would like to extend our probabilistic model to incorporate non-ideal channel effects—such as external interference—to accurately reflect real-world conditions. Moreover, we aim to broaden our scheduling framework by devising algorithms that support transmissions even when deadlines do not align with super-frame boundaries. Finally, a key future direction is to integrate downlink transmissions into our super-frame structure, closing the loop for sensor-actuator communication.

References

1 W. Ji, T. Ebrahimi, Z. Li, J. Yuan, D. O. Wu, and Y. Xin, "Guest editorial: Emerging visual iot technologies for future communications and networks," *IEEE Wireless Communications*, vol. 28, no. 4, pp. 10–11, 2021.

2 M. Jouhari, N. Saeed, M.-S. Alouini, and E. M. Amhoud, "A survey on scalable lorawan for massive iot: Recent advances, potentials, and challenges," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 3, pp. 1841–1876, 2023.

3 A. Askhedkar, B. Chaudhari, and M. Zennaro, "Hardware and software platforms for low-power wide-area networks," in *LPWAN technologies for IoT and M2M applications*. Elsevier, 2020, pp. 397–407.

4 B. Al Homssi, A. Al-Hourani, K. Magowe, J. Delaney, N. Tom, J. Ying, H. Wolf, S. Maselli, S. Kandeepan, K. Wang *et al.*, "A framework for the design and deployment of large-scale lpwan networks for smart cities applications," *IEEE Internet of Things Magazine*, vol. 4, no. 1, pp. 53–59, 2020.

5 D. D. Olatinwo, A. Abu-Mahfouz, and G. Hancke, "A survey on lpwan technologies in wban for remote health-care monitoring," *Sensors*, vol. 19, no. 23, p. 5268, 2019.

6 D. Hernandez, G. Peralta, L. Manero, R. Gomez, J. Bilbao, and C. Zubia, "Energy and coverage study of lpwan schemes for industry 4.0," in *2017 IEEE International Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM)*. IEEE, 2017, pp. 1–6.

7 A. Jain, M. A. Haque, A. Saifullah, and H. Zhang, "Burst-mac: A mac protocol for handling burst traffic in lora network," in *2024 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2024, pp. 148–160.

8 J. C. Liando, A. Gamage, A. W. Tengourtius, and M. Li, "Known and unknown facts of lora: Experiences from a large-scale measurement study," *ACM Trans. Sen. Netw.*, vol. 15, no. 2, Feb 2019.

9 R. Gilson and M. Grudsky, "Lorawan capacity trial in dense urban environment," machineQ and Semtech, Tech. Rep., 2017. [Online]. Available: https://info.semtech.com/hubfs/machineQ_LoRaWAN_Capacity_Trial-2.pdf

10 K. Banti, I. Karampelia, T. Dimakis, A.-A. A. Boulogeorgos, T. Kyriakidis, and M. Louta, "Lorawan communication protocols: A comprehensive survey under an energy efficiency perspective," *Telecom*, vol. 3, no. 2, pp. 322–357, 2022.

11 B. Vejlgaard, M. Lauridsen, H. Nguyen, I. Z. Kovács, P. Mogensen, and M. Sorensen, "Coverage and capacity analysis of sigfox, lora, gprs, and nb-iot," in *2017 IEEE 85th vehicular technology conference (VTC Spring)*. IEEE, 2017, pp. 1–5.

12 A. Gamage, J. Liando, C. Gu, R. Tan, M. Li, and O. Seller, "Lmac: Efficient carrier-sense multiple access for lora," *ACM Trans. Sen. Netw.*, vol. 19, no. 2, 2023.

13 A. Gamage, G. de Guillebon, M. Li, M. Luis, and O. Seller, "Enabling csma for lorawan," LoRa Alliance, USA, Technical Report TR-013, September 2023.

14 F. Yu, X. Zheng, L. Liu, and H. Ma, "Loradar: An efficient lora channel occupancy acquirer based on cross-channel scanning," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*. IEEE Press, 2022, p. 540–549. [Online]. Available: https://doi.org/10.1109/INFOCOM48880.2022.9796845

15 J. M. Finochietto, O. Dieng, D. Mosse, and R. M. Santos, "Adding empirical real-time guarantees to lorawan," in *Proceedings of the 31st International Conference on Real-Time Networks and Systems*, 2023, pp. 99–107.

16 S. Fahmida, V. P. Modekurthy, D. Ismail, A. Jain, and A. Saifullah, "Real-time communication over lora networks," in *2022 IEEE/ACM Seventh International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, 2022, pp. 14–27.

17 S. Fahmida, A. Jain, V. P. Modekurthy, D. Ismail, and A. Saifullah, "Rtpl: A real-time communication protocol for lora network," *ACM Transactions on Embedded Computing Systems*, vol. 24, no. 1, pp. 1–31, 2024.

18 L. A. T. Committee, "LoRaWAN™ 1.1 Specification," https://lora-alliance.org/resource_hub/lorawan-specification-v1-1/, LoRa Alliance, October 2017, final Release, accessed: 2025-05-05.

19 Y.-H. Wei, Q. Leng, S. Han, A. K. Mok, W. Zhang, and M. Tomizuka, "Rt-wifi: Real-time high-speed communication protocol for wireless cyber-physical control applications," in *2013 IEEE 34th Real-Time Systems Symposium*. IEEE, 2013, pp. 140–149.

20 A. Saifullah, Y. Xu, C. Lu, and Y. Chen, "Real-time scheduling for wirelesshart networks," in *2010 31st IEEE Real-Time Systems Symposium*. IEEE, 2010, pp. 150–159.

21 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. 10 812–10 823, 2019.

22 E. Felemban and E. Ekici, "Single hop ieee 802.11 dcf analysis revisited: Accurate modeling of channel access delay and throughput for saturated and unsaturated traffic cases," *IEEE Transactions on Wireless Communications*, vol. 10, no. 10, pp. 3256–3266, 2011.

23 G. Bianchi, "Performance analysis of the ieee 802.11 distributed coordination function," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 3, pp. 535–547, 2000.

24 S. K. Nobar, K. A. Mehr, and J. M. Niya, "Comprehensive performance analysis of ieee 802.15.7 csma/ca mechanism for saturated traffic," *J. Opt. Commun. Netw.*, vol. 7, no. 2, pp. 62–73, Feb 2015. [Online]. Available: https://opg.optica.org/jocn/abstract.cfm?URI=jocn-7-2-62

25 G. Bianchi, "Performance analysis of the ieee 802.11 distributed coordination function," *IEEE Journal on selected areas in communications*, vol. 18, no. 3, pp. 535–547, 2000.

26 P. Robyns, P. Quax, W. Lamotte, and W. Thenaers, "A multi-channel software decoder for the lora modulation scheme," in *Proceedings of the International Conference on Internet of Things, Big Data and Security (IoTBDS)*. SciTePress, 2018.

27 P. Xie, Y. Li, Z. Xu, Q. Chen, Y. Liu, and J. Wang, "Push the limit of lpwans with concurrent transmissions," in *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications*, 2023, pp. 1–10.

28 Z. Xu, J. Luo, Z. Yin, S. Wang, C. Chen, J. Lin, R. Xiong, and T. He, "X-mac: Achieving high scalability via imperfect-orthogonality aware scheduling in lpwan," in *2022 IEEE 30th International Conference on Network Protocols (ICNP)*, 2022, pp. 1–11.

29 Y. Wang, F. Zhang, X. Zheng, L. Liu, and H. Ma, "Decoding lora collisions via parallel alignment," *ACM Transactions on Sensor Networks*, vol. 19, no. 3, pp. 62:1–62:25, 2023.

30 X. Xia, Q. Chen, N. Hou, and Y. Zheng, "Hylink: Towards high throughput lpwans with lora compatible communication," in *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems*, 2023, pp. 578–591.

31 C. Shao, O. Muta, Q. Du, K. R. Dandekar, and X. Wang, "Multiple access in large-scale lorawan: Challenges, solutions, and future perspectives," *IEEE Consumer Electronics Magazine*, vol. 13, no. 5, pp. 36–46, 2024.

32 N. Hou, X. Xia, and Y. Zheng, "Cloaklora: A covert channel over lora phy," *IEEE/ACM Transactions on Networking*, vol. 31, no. 3, pp. 1159–1172, 2023.

33 C. Shao, O. Muta, W. Wang, and W. Lee, "Toward ubiquitous connectivity via lorawan: An overview of signal collision resolving solutions," *IEEE Internet of Things Magazine*, vol. 4, no. 4, pp. 114–119, 2021.

34 K. Yang, Y. Chen, X. Chen, and W. Du, "Link quality modeling for lora networks in orchards," in *Proceedings of the 22nd International Conference on Information Processing in Sensor Networks (IPSN)*, 2023, pp. 27–39.

35 Z. Xu, P. Xie, J. Wang, and Y. Liu, "Ostinato: Combating lora weak links in real deployments," in *2022 IEEE 30th International Conference on Network Protocols (ICNP)*, 2022, pp. 1–11.

36 C. Shao and O. Muta, "When lorawan meets csma: Trends, challenges, and opportunities," *IEEE Internet of Things Magazine*, vol. 7, no. 1, pp. 90–96, 2024.

37 G. Gaillard and C. Pham, "Canl lora: Collision avoidance by neighbor listening for dense lora networks," in *2023 IEEE Symposium on Computers and Communications (ISCC)*, 2023, pp. 1293–1298.

38 A.-U.-H. Ahmar, E. Aras, T. D. Nguyen, S. Michiels, W. Joosen, and D. Hughes, "Design of a robust mac protocol for lora," *ACM Transactions on Internet of Things*, vol. 4, no. 1, pp. 3:1–3:25, 2023.

39 I. of Electrical and E. Engineers, "IEEE Standard for Information Technology-Local and Metropolitan Area Networks-Specific Requirements-Part 15.1: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs)," IEEE Std 802.15.1-2005 (Revision of IEEE Std 802.15.1-2002), IEEE, June 2005, published 14 June 2005.

40 ——, "IEEE Standard for Information Technology-Telecommunications and Information Exchange Between Systems-Local and Metropolitan Area Networks-Specific Requirements-Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," IEEE Std 802.11-2007 (Revision of IEEE Std 802.11-1999), IEEE Computer Society, June 2007, published 12 June 2007.

41 Semtech, "Sx1301 lora core digital baseband chip," https://www.semtech.com/products/wireless-rf/lora-core/sx1301, 2024, accessed: 2024-10-04.

42 S. R. T. C. Ltd., "Rak831 lora gateway module," https://www.rakwireless.com/en-us/products/rak831, Shenzhen RAKwireless Technology Co. Ltd., 2020, accessed: 2025-05-05.

43 R. S. et al., "Bsma: Scalable lora networks using full duplex gateways," in *Proceedings of the 28th Annual International Conference on Mobile Computing and Networking*, 2022, pp. 676–689.

44 S. Yu, X. Xia, Z. Zhang, N. Hou, and Y. Zheng, "Fdlora: Tackling downlink-uplink asymmetry with full-duplex lora gateways," in *Proceedings of the 22nd ACM Conference on Embedded Networked Sensor Systems (SenSys)*. Hangzhou, China: ACM, November 4-7 2024.

45 A. Gamage, "Optimizing spectral utilization of lpwans," Doctoral thesis, Nanyang Technological University, 2023.

46 I. of Electrical and E. Engineers, "IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," IEEE Std 802.11-1997, 1997, pages 1–445.

47 G. Chiaselotti, M. I. Gualtieri, and P. Pietramala, "Minimizing the makespan in nonpreemptive parallel machine scheduling problem," *Journal of Mathematical Modelling and Algorithms*, vol. 9, pp. 39–51, 2010.

48 V. P. Modekurthy, A. Saifullah, and S. Madria, "Distributedhart: A distributed real-time scheduling system for wirelesshart networks," in *2019 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2019, pp. 216–227.

49 H. Automation, "Heltec wireless stick lite v3: Esp32s3 + sx1262 lora node," https://heltec.org/project/wireless-stick-lite-v2/, Heltec Automation, 2025, accessed: 2025-05-05.

50 J. A. Donenfeld, "Wireguard: Next generation kernel network tunnel," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2017, available: https://www.ndss-symposium.org/ndss2017/ndss-2017-programme/wireguard-next-generation-kernel-network-tunnel/.

51 Semtech, "Lora gateway project: Driver/hal for semtech sx1301-based concentrator boards," https://github.com/Lora-net/lora_gateway, Semtech, 2020, accessed: 2025-05-05.

52 S. Tong, J. Wang, J. Yang, Y. Liu, and J. Zhang, "Citywide lora network deployment and operation: Measurements, analysis, and implications," in *Proceedings of the 21st ACM Conference on Embedded Networked Sensor Systems*, 2024.