

RTPL: A Real-Time Communication Protocol for LoRa Network

SEZANA FAHMIDA*, Wayne State University, USA

AAKRITI JAIN*, Wayne State University, USA

PRASHANT MODEKURTHY*, University of Nevada Las Vegas, USA

DALI ISMAIL, State University of New York, USA

ABUSAYEED SAIFULLAH, Wayne State University, USA

The industrial Internet of Things (IIoT) is prominently emerging in applications of large-scale and wide-area applications, such as oilfield management, smart grid management, real-time equipment monitoring, and integration of traffic management systems for smart cities. Relying on short-range wireless technologies (e.g., WirelessHART and ISA100.11a), traditional wireless solutions for industrial automation find it challenging to support the expansive scale of today's IIoT. To address this limitation, we propose to adopt LoRaWAN, a prominent low-power wide-area network technology, for industrial automation. LoRaWAN for industrial automation poses some unique challenges. The fundamental building blocks of any industrial automation system are feedback control loops that largely rely on real-time communication. LoRaWAN traditionally adopts a simple protocol based on ALOHA with no collision avoidance or Listen Before Talk with Clear Channel Assessment and Random Backoff mechanisms to minimize energy consumption, which are less suitable for real-time communication. Existing real-time protocols for short-range technologies cannot be applied to a LoRaWAN network due to its unique characteristics such as asymmetry between downlink and the uplink spectrum, predefined modes (or classes) of operation, and concurrent reception through orthogonal spreading factors. In this paper, we address these challenges and propose RTPL- a Real-Time communication Protocol for LoRaWAN networks. RTPL is a low-overhead and conflict-free communication protocol allowing autonomous real-time communication of low-energy devices and exploits LoRa's capability of parallel communication. We implement our approach on LoRa devices and evaluate through both physical experiments and extensive simulations. All results show that RTPL achieves on average 75% improvement in real-time performance without sacrificing throughput or energy compared to traditional LoRaWAN.

CCS Concepts: • **LoRaWAN**; • **LoRa**; • **Low Power Wide Area Networks**; • **Real-Time**;

ACM Reference Format:

Sezana Fahmida, Aakriti Jain, Prashant Modekurthy, Dali Ismail, and Abusayeed Saifullah. 2024. RTPL: A Real-Time Communication Protocol for LoRa Network. *ACM Trans. Embedd. Comput. Syst.* 1, 1 (October 2024), 31 pages. <https://doi.org/XXXXXXXX.XXXXXXX>

1 INTRODUCTION

Internet of Things (IoT) is facilitating the transformation of industrial automation including process control and smart manufacturing through Industrial Internet of Things (IIoT). *Industrial automation* uses sensors and actuators connected to a low-bandwidth network to manage, monitor, and control production or manufacturing processes. By integrating machines, cloud computing, analytics, and human expertise, Industrial IoT (IIoT) enhances the efficiency and promotes sustainable production.

*Co-first author

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2024 Association for Computing Machinery.

1539-9087/2024/10-ART \$15.00

<https://doi.org/XXXXXXXX.XXXXXXX>

Today, industrial IoT and Cyber-Physical Systems (CPS) are emerging in large-scale and wide-area applications. For example, the East Texas Oil-field extends over an area of $74 \times 8 \text{ km}^2$ requiring tens of thousands of sensors and actuators for automated management [30]. Emerson is targeting to deploy 10,000 nodes (sensors and actuators) for managing an oil-field in Texas [17]. Today, wireless solutions for industrial automation are based on WirelessHART [18] and ISA100.11a [10] that depend on traditional short-range wireless technologies based on IEEE 802.15.4. To cover a large area with numerous devices, they form multi-hop mesh networks at the expense of energy, cost, and complexity, posing a big challenge to support the scale of today's IIoT. In this paper, we propose to adopt the *Low-Power Wide-Area Network (LPWAN)* technologies for industrial automation. As an emerging IoT technology, LPWAN enables low-power (milliwatts) wireless devices to transmit at low data rates (kbps) over long distances (kms) using narrow band (kHz), thereby obviating the need of multi-hop and allowing the devices to directly communicate with the control node (gateway). Recently, multiple LPWAN technologies such as LoRa (Long Range) [12], SigFox [15], DASH7 [1], LTE Cat M1 [2], and SNOW (Sensor Network Over White spaces) [52] have appeared. Many of them (e.g., LoRa, SigFox, SNOW) allow numerous devices to concurrently communicate with the gateway, providing high scalability.

IIoT realization through LPWAN can greatly benefit the industrial automation solutions like pipeline management [47], silo level, environmental and cold chain control. In such applications, pipelines (that can be hundreds of miles long), silos, tanks, and plants are positioned far from the central operations center, at inconvenient or hazardous locations in difficult terrain or offshore. Thanks to their multi-kilometer range and deep penetration capability, LPWANs can be an attractive solution for communications from massive, granular data points of geographically dispersed and/or structurally dense industrial campuses like oil fields, refineries and process plants. Compared to industrial mesh solutions (e.g., WirelessHART, ISA 100.11a), they can be implemented without complex network configuration and at a fraction of both device and operational costs. In a 2018 survey on 311 industries conducted by ON World and the International Society of Automation (ISA), as shown in Fig. 1, 57% of industrial IoT professionals reported that they were researching or developing LPWAN solutions [31]. Thus, LPWANs will soon be disrupting the IIoT landscape.

To avoid the cost of licensed band and infrastructure (usually unavailable in remote locations where process industries are typically located), we consider non-cellular LPWANs in the free ISM band. Specifically, we consider LoRa, which is widely considered as an LPWAN leader and is commercially available all around the globe with more than 600 known use cases and over 50 million devices deployed [12]. Industry analyst ABI research projects that more than 50% of all LPWAN connections will be based on LoRa/LoRaWAN by 2026 as it is flexible for both outdoor and indoor cases [41].

Adopting LoRa for industrial automation poses some challenges as it was not originally targeted for such applications. The fundamental building blocks of any industrial automation system are feedback control loops that largely rely on real-time communication between sensors and actuators [37]. For example, tanks in oil-fields need real-time monitoring and control to avoid overflow.

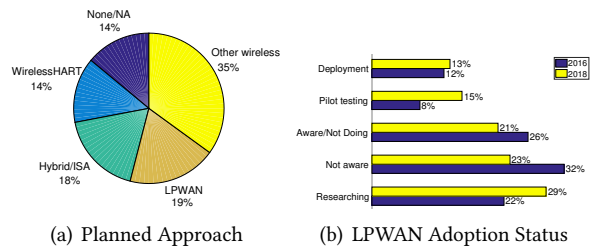


Fig. 1. LPWAN adoption trend in industrial automation [19, 31]

However, LoRa was predominantly developed for independent uplink and downlink communications. Enabling closed-loop communication under severe energy constraints of the nodes is quite challenging. To minimize the energy consumption of devices, LoRa usually adopts a simple MAC (media access control) protocol known as *LoRaWAN (Long Range Wide-Area Network)*, which is based on ALOHA with no collision avoidance. Some variants of LoRaWAN adopt Listen Before Talk (LBT) with Clear Channel Assessment and Random Backoff mechanism [46] to minimize collision probability. Such protocols are naturally *unsuited* for real-time communication as they introduce variability in uplink transmissions, leading to large and sometimes unbounded latency.

While energy-efficiency is a requirement and challenge in LPWANs, it becomes more complicated when combined with real-time requirement. Specifically, their communications have to be minimal which makes real-time communication extremely challenging. On the other hand, real-time communications can benefit from massive concurrent communication of the devices with the gateway. Existing real-time communication protocols developed for short-range technologies cannot exploit the massive concurrent communication and hence cannot be adopted for LoRaWAN. The asymmetry between the downlink and uplink spectrum in a LoRaWAN network (in regions like North America, China, Australia) also makes it difficult to adopt traditional real-time communication protocols. Existing real-time communication protocols are mostly centralized and also assume that the gateways receive from one node at a time, gateways can either transmit a packet or receive a packet at a time, but not both, and uplink and downlink communications happen on the same spectrum, which are ineffectual under LoRaWAN.

Government agencies impose restrictions at the national level to ensure accessibility and safety. Some countries impose Duty Cycle limitations on LoRa devices which limits the transmission time of a device. For instance, Europe has a duty cycle limitation of 1% on the devices running ALOHA access protocol. Duty cycle limitations can significantly reduce the available airtime for transmitting critical data and hence imposes a challenge in adopting LoRaWAN for real-time communication.

In this paper, we address the above challenges and propose **RTPL**, a Real-Time communication Protocol for LoRaWAN networks that can ensure real-time guarantees of end-to-end communication. RTPL closes the control loops over asymmetric spectrum allocation for uplink and downlink communication and exploits LoRa's capability of parallel communication. It is a low-overhead real-time MAC protocol allowing autonomous communication of low-energy devices. Furthermore, we present a *schedulability analysis* for RTPL. In real-time process control applications, schedulability analysis is widely used to determine, both during the design time and for online admission control, if a given set of control loops/flows can be scheduled with a given scheduling algorithm such that none of the loops in the set miss their deadlines. It facilitates a proactive network management by enabling network manager to plan ahead and adjust workloads. In summary, the paper makes the following contributions:

- (1) We propose an autonomous real-time scheduling framework for LoRaWAN called **RTPL**, a low-overhead real-time MAC protocol allowing autonomous communication of low-power devices under large deployment.
- (2) We exploit the concurrent communication capabilities of LoRa to design RTPL and extend it to handle frequency hopping, duty-cycling, link failures, and network and workload dynamics.
- (3) We present a Schedulability Analysis for RTPL.
- (4) We implement RTPL on LoRa devices and evaluate through both physical experiments and large scale simulations. All results show that RTPL achieves on average 75% improvement in real-time performance without sacrificing throughput or energy compared to traditional LoRaWAN.

The rest of the paper is organized as follows. Section 2 presents an overview of LoRaWAN. Section 3 describes the system model. Section 4 reviews related work. Section 5 motivates RTPL. Section 6 presents RTPL. Section 7, 8 and 9 presents implementation, experiment results, and simulation results, respectively. Section 10 concludes the paper.

2 AN OVERVIEW OF LORAWAN

LoRaWAN is a pioneering LPWAN technology for connecting sensors to the Cloud. In the LoRa stack, LoRaWAN is refer to as the MAC layer protocol and LoRa is refer to as the physical layer (PHY) modulation technique. A typical LoRaWAN architecture (shown in Fig. 2) consists of three parts: gateway, end-devices i.e., *nodes*, and a network server. Nodes are sensors/actuators that directly communicate with the gateway on a single-hop wireless link. LoRaWAN divides the available spectrum into multiple *uplink* and *downlink* channels. Nodes send data to the gateway on the uplink channels and the gateway responds on the downlink channels. Multiple gateways communicate with the network server via a local LAN/the Internet.

The LoRa radios exploit Chirp Spread Spectrum (CSS) modulation for communication. Spreading the signal over a wide bandwidth makes it less susceptible to noise and interference. A LoRa transceiver has five runtime-adjustable transmission parameters: transmission power, carrier frequency (channel), *spreading factor* (*SF*), bandwidth (BW), and coding rate (CR). These parameters influence the transmission duration, energy consumption, reliability, and range. In North America, LoRa defines 64 uplink channels of 125 kHz bandwidth (902.3-914.9 MHz) in 200 kHz increments and 8 downlink channels of 500 kHz bandwidth (923.3 MHz-927.5 MHz) in 600 kHz increments. Under FCC, there is no duty-cycle requirement on spectrum usage for LoRa in the US but there is a maximum dwell time of 400ms per channel on uplinks [3, 40]. CR is the amount of Forward Error Correction (FEC) applied to the message to protect it against burst interference. It ranges between 1 and 4. SF determines the number of bits encoded in a symbol and ranges from 6 to 12. A higher spreading factor increases the Signal to Noise Ratio (SNR) and, therefore, receiver sensitivity and the signal range. Higher the SF implies lower bit rate, and hence, the longer the communication time. LoRa signal data rate is given by $SF * \frac{4/(4+CR)}{2^{SF/BW}}$. Depending on the SF and bandwidth, LoRa achieves data rates between 0.3 kbps and 27 kbps. Semtech claims that the SFs in LoRa are orthogonal [13] i.e., concurrent transmissions with different SF on the same channel do not interfere with each other and can be successfully decoded.

In LoRaWAN, the nodes are categorized into three classes. **Class A** nodes are allowed for bidirectional communication where each node's uplink transmission is followed by two short downlink receive windows. In the Class A mode, nodes locally schedule transmissions based on their own communication needs with a small variation based on a random time basis (ALOHA-type protocol). This Class A operation is lowest power-consuming and intended for applications that only require downlink communication from the gateway shortly after a node's uplink transmission. Downlink communications from the gateway at any other time must wait until the next scheduled uplink. **Class B** nodes are also allowed for bidirectional communication, where they allow for more receive slots. In addition to the Class A receive windows, Class B nodes open extra receive windows at pre-scheduled times. To resolve clock drifts, nodes' synchronizes their time with the gateway through time synchronizing beacons. **Class C** nodes have continuously open receive windows,

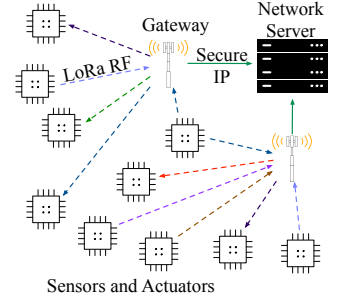


Fig. 2. The LoRa network architecture.

only closed when transmitting. Class C mode consumes higher power but offers the lowest latency communication from the gateway.

3 SYSTEM MODEL

We consider an LPWAN based on LoRaWAN. A LoRaWAN node is equipped with one half-duplex radio, enabling transmission or reception of a packet at a time, but not both. In a LoRaWAN network, a channel and a SF together is called a *communication path*. We define an *uplink communication path* (UCP) as an element in the set of unique combinations of uplink channels and spreading factors on which a gateway can simultaneously receive packets. Similarly, we define a *downlink communication path* (DCP) as an element in the set of unique combinations of downlink channels and spreading factors on which a gateway can simultaneously transmit packets.

LoRaWAN technical specification from Semtech mentions that two communication paths with different channels or different SFs are inherently *orthogonal* to each other, implying that concurrent transmissions on the same channel and different SFs do not interfere with each other [14]. Some recent studies have found through experiments that different SFs in LoRa are quasi-orthogonal, and signals can be decoded only if specific relationships between their power are held [23, 39, 42]. Currently available Commercial Off-The-Shelf (COTS) gateways may have limitations and might not fully support SF orthogonality. However, gateways that support SF orthogonality will likely be available in the future. In this paper, we consider concurrent communications on different SFs on the same channel are orthogonal as per the LoRa specification.

A LoRa gateway has $(1 + m')$ radios. One radio enables concurrent reception on m uplink orthogonal communication paths (UCPs), and all other radios enable concurrent transmissions on m' downlink orthogonal communications paths (DCPs). UCPs are denoted as c_1, c_2, \dots, c_m and DCPs are denoted as $\delta_1, \delta_2, \dots, \delta_{m'}$. All UCPs use a BW of 125 khz, and all DCPs use a BW of 500 kHz. The number of orthogonal SFs available in the network is denoted by η . The spreading factor and channel used for the j^{th} UCP c_j is given by $\Phi(c_j)$ and $\zeta(c_j)$, respectively. Similarly, the spreading factor and channel used for a DCP δ_j is given by $\Phi(\delta_j)$ and $\zeta(\delta_j)$, respectively.

We consider an LPWAN with n wireless control loops $\ell_1, \ell_2, \dots, \ell_n$, where ℓ_i is the control loop between a sensor node s_i and actuator a_i through a controller at the gateway. The controller is connected via a wi network to the gateway, and the gateway acts as a bridge between the controller and the wireless network. In industrial automation applications, a sensor node samples a plant (or a process) state and sends data to the controller (at the gateway) along a UCP. Upon receiving a plant's state information, the controller determines control commands and sends them to the actuators along a DCP. The sampling period of control loop ℓ_i is denoted by p_i . The deadline d_i of control loop ℓ_i equals its period p_i (i.e., $p_i = d_i$). This means that the end-to-end communication between sensor s_i and actuator a_i has to be completed within p_i time units. The actuator a_i must receive the control command within p_i time units after the generation of a sample. The actuator applies the control command to modify the state of the plant. The objective of RTPL is to meet the deadlines of the control loops. A control loop ℓ_i is *schedulable* if it meets its deadline d_i for all of its packets. A set of control loops is called *schedulable*, if every control loop of the set is schedulable.

4 RELATED WORK

Real-time scheduling for wireless network was studied widely for short range technologies such as IEEE 802.15.4 standard [43, 44, 51, 56, 64], Bluetooth Low Energy [26, 48] and WiFi [9]. However, these approaches cannot be applied to a LoRaWAN network due to its unique characteristics such as (1) asymmetric communication paths between uplink and downlink, (2) nodes' restriction to operate in one of the three modes (A, B, or C) with severe energy constraints, and (3) concurrent reception capability by exploiting orthogonal SFs.

Existing work on LoRa/ LoRaWAN mostly focuses on improving performance [28, 59], energy consumption and coverage [27, 29], scalability [39, 41, 50], and packet collision resolution [55, 57, 60–63]. Several studies [45, 49] have investigated the applicability of LoRaWAN in industrial settings. However, these studies have not introduced any real-time communication solutions or protocols for packet transmissions and timing guarantees for LoRaWAN; instead, they have analyzed the potential of LoRaWAN for industrial use. Some works have applied pre-existing LoRaWAN techniques for specific real-time monitoring applications such as road traffic monitoring [54] or water-quality monitoring systems [36]. Our work does not focus on building real-time monitoring systems. Rather we propose a *Real-Time MAC Protocol*, which allows autonomous communication of low-energy LoRaWAN devices.

The potential of LoRa/LoRaWAN in real-time communication and control was explored in recent works [24, 32, 35, 58, 65]. The work in [65] provides a time-slotted MAC layer on top of LoRaWAN to improve reliability but does not propose any real-time protocol. While the work in [35] considers both real-time and non real-time traffic in a LoRaWAN network, it evaluates through simulations without any implementation on LoRa devices. It provides a general overview of how time slots should be structured based on the chosen SF and channel but does not propose a real-time algorithm or slot scheduling mechanism to meet the real-time requirements of nodes. Besides, it relies on the offline computation of slots without giving any mechanism to assign these slots to the flows to ensure real-time communication. Also, it considers a simplified model of traffic where all nodes in the network operate on the same transmission interval and does not consider the scheduling of downlink actuator commands. As already pointed out in [32], this approach is not applicable to a real industrial IoT network for real-time communication.

While the approach in [32] considers assigning time slots for uplink communication, it does not consider any sort of real-time scheduling for downlink communication. It considers time slots of an equal length while a LoRa transmission's slot length depends on spreading factor. For example, the time slot length for a transmission on spreading factor 7 is two times that for a transmission on spreading factor 6; similarly, the time slot length for a transmission on spreading factor 8 is two times that for a transmission on spreading factor 7, and so on. Also, the approach leverages node grouping based on the number of hops in the network requiring a LoRa node to have the capability of forwarding a packet to another LoRa node while the LoRa nodes are not capable of direct peer-to-peer communication. The work in [58] has also been proposed by the same authors based on the assumption that a LoRa node is capable of forwarding a packet to another LoRa node. Finally, the evaluation of [32, 58] does not show any result to demonstrate its real-time performance.

Enabling real-time communication upon closing the loop over a LoRa network is quite challenging due to asymmetric spectrum between uplink and downlink and severe energy constraints of the nodes. Closed-loop communication in LoRa networks has been studied in [24] to enable event-triggered control. But its proposed co-design does not consider real-time communication. It relaxes the energy constraints as the actuators need to always remain in continuous listen mode, and also does not exploit LoRa's capability of concurrent reception on orthogonal SFs. Most importantly, in its approach, the nodes contend for data transmission slots before sending the data, which can lead to collisions and unpredictable latency. In contrast, our approach adopts a time-triggered model and conflict-free communication, thereby enabling predictable latency and real-time communication. Our approach exploits LoRa's capability of parallel reception and preserves energy as both sensors and actuators only wake up to send or receive data.

5 FEASIBILITY OF CLOSED LOOP REAL-TIME COMMUNICATION WITH LORAWAN

LoRaWAN facilitates low-latency communication. However, the use of ALOHA-based protocol in LoRaWAN for uplink communication that causes significant packet collisions, unpredictable

latencies, and deadline misses. Furthermore, it introduces challenges in closing the loop. As described in Section 4, there is no existing work on closed-loop conflict-free MAC protocol for real-time communication that consider LoRaWAN's characteristics. Hence, in this section, we experimentally investigate the following: (1) Is real-time communication feasible between sensors and gateway? (2) Is closed-loop real-time communication feasible in LoRa? and (3) Can LoRaWAN adopt existing wireless sensor-actuator network (WSAN) MAC protocols?

5.1 Can LoRaWAN Ensure Real-Time Guarantees for Uplink Communication?

We conducted an experiment to demonstrate the LoRaWAN performance in a real-time high traffic scenario. High traffic scenarios with low periods are common for industrial wireless control applications. Implicit deadline systems (deadline = period) and constrained deadline (deadline < period) systems are the most common system models. In a monitoring/control application, delivering a previous packet after a new packet is generated is irrelevant. Hence, a packet must be delivered before the next packet is generated, thereby making the period an implicit deadline. To model this behavior, we use an implicit deadline system in our paper.

We ran the experiment with 4 Raspberry Pi 3 with LoRa HAT from Dragino [4] as LoRa sensor nodes and one Dragino LG308 LoRaWAN gateway with The Things Stack network server [16]. To replicate a high traffic scenario we choose a period and deadline of 7s – the sum of transmission time and length of reception windows (5s) under the things stack [16]. The nodes transmitted 10-byte packets to the gateway. All nodes used the same channel to emulate a dense deployment. The nodes used a 5 second receive window delay as per the recommendations from The Things Stack. We do not change the reception windows to increase the chance of successfully receiving an acknowledgment. We measured the latency between packet generation at the sensor and packet reception at the gateway. We compared the measured latency with the deadline to observe the number of packets that meet the deadline.

Fig. 3 shows the number of packets out of 30 that were successfully received before the deadline by the gateway from each sensor. This result shows that 100% of packets missed the deadline for each sensor. The results thus show that under high traffic scenarios, LoRaWAN is prone to collisions and is not suitable for time-sensitive applications/ real-time communication.

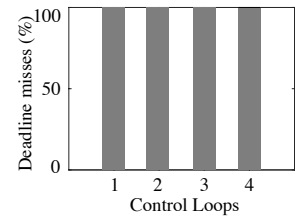


Fig. 3. Percentage of deadline misses.

5.2 Is Closed-loop Real-Time Communication Feasible under LoRaWAN?

Feedback control loops are the fundamental building blocks in many IIoT and industrial CPS applications [33]. In a feedback loop, a sensor observes the state of the plant and transmits the observed state to a controller residing at the gateway. Upon receiving the observed state, the controller generates a control command and transmits it to the actuator. The current LoRaWAN MAC is not developed for handling such dependencies between uplink and downlink communication. As a first step, we try to see the feasibility of closing the loop using Class B for downlink communication while the uplink communications happen in Class A mode. Scheduling downlink messages using Class B on predetermined time slots would require us to estimate latency of uplink communication.

We perform experiments to demonstrate the latency distribution of packets in uplink communication. We show that predicting an upper bound of latency (not the minimum possible latency) is integral to a real-time communication protocol and is highly challenging. We used a similar experimental setup to that in the previous section. For this experiment, we define latency of a packet as the time taken to transmit a packet to the gateway and receive an acknowledgment.

Fig. 4 shows the cumulative distribution function (CDF) of packet latency for 30 packets using LoRaWAN. We observed that the latency of the packet sent from the same node varies significantly over a wide range. When the packet was successful in the first attempt, the latency of the packet was under 0.8 s. However, if the packet was not successfully received on the first attempt, nodes waited for their 5 s receive window, and then retransmitted the packet. After one retransmission, packets incur a latency of 7.8 s. We observed that in an outdoor experiment, 94% of the packets were successfully received after one retransmission while 6% of the packets were received on the first transmission. In this experiment, we observed an average latency of 7.4 s and a maximum latency of 7.8 s.

In industrial internet-of-things, downlink communication on a control loop (from a gateway to an actuator) follows an uplink communication on the same control loop (from sensor to gateway). To schedule a downlink communication, the gateway may use a ping slots for communication. However, to schedule a ping slot, the gateway requires an upper bound on the uplink communication latency prior, to scheduling dissemination. The result from our experiment shows that estimating an effective upper bound of uplink latency is quite challenging as uplink communication under LoRaWAN MAC is prone to collisions, leading to unpredictability in latency. Therefore, it is challenging to schedule the downlink transmissions using Class B. Using Class C for downlink is also not a practical option for energy constrained LoRaWAN nodes as Class C mode has extremely high energy overhead [5]. This result demonstrates the need for a new approach to close feedback loops with real-time communication.

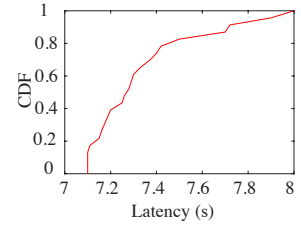


Fig. 4. Latency variation in LoRaWAN.

5.3 Can we Adopt Existing WSN MAC Protocols for LoRa?

Real-time communication protocols have been proposed for industrial WSN based on short-range wireless technologies like WirelessHART [18] and ISA100.11a [10]. These protocols are mostly centralized and depend on table-driven scheduling based on Time Division Multiple Access (TDMA). These protocols make several assumptions like (1) gateways receive from one node at a time, (2) gateways can either transmit a packet or receive a packet at a time, but not both, and (3) uplink and downlink communications happen on the same spectrum. These assumptions are ineffectual under LoRaWAN. In regions like North America, China, and Australia, a LoRa gateway uses asymmetric uplink and downlink spectrum in terms of band, bandwidth, and spreading factors. A LoRa gateway can receive multiple packets simultaneously, and it uses multiple radios to transmit and receive packets concurrently. Currently, the number of available downlink channels for LoRaWAN is significantly less than uplink, specifically in the US. This asymmetry of uplink and downlink resources makes it difficult to adopt traditional real-time scheduling techniques in LoRaWAN. One naive way of tackling this challenge is to use the same number of uplink and downlink channels. However, this approach leaves the uplink resources mostly under-utilized. Since existing real-time communication protocols for industrial WSN assume uplink and downlink communications happen on the same spectrum, they are not directly applicable to LoRaWAN. Enabling real-time communication over LoRa for industrial automation needs a new approach.

6 DESIGN OF RTPL

In this section, we design RTPL by developing an autonomous real-time scheduling framework for LoRaWAN. It is designed by exploiting the concurrent communication capabilities of LoRa. It also handles frequency hopping, duty-cycling, link failures, network and workload dynamics. For

simplicity in explanation, we initially describe the design of RTPL when the number of UCPs and DCPs are the same. Later, we extend the design of RTPL to unequal number of UCPs and DCPs.

6.1 Deciding the Scheduling Approach: Global or Partitioned?

The current LoRaWAN adopts ALOHA or Listen Before Talk (LBT) and is naturally unsuited for industrial automation. We adopt a Time Slotted Channel Hopping (TSCH) like approach which can provide predictable latency (and minimize packet collisions) and is prefer in industrial automation [10, 18]. While time-slotted communication needs time synchronization, in RTPL the gateway transmits a piggy-backed time stamp on each downlink communication, i.e., acknowledgments to sensors and control communication to actuators. The nodes (sensors and actuators) synchronize their local clocks with the gateway's clock. This approach reduces the need for additional transmissions for synchronization and decreases energy consumption. However, it may lead to time synchronization due to clock drift at the node. To account for synchronization errors during sleep, the time slot calculation includes the safety margin and packet transmission time on a selected SF. Safety margins are predetermined by the network manager during the deployment phase, specifically after testing clock drift at nodes between sampling periods.

We use an online scheduler (discussed in Section 6.2), wherein sensors and actuators locally generate the communication schedule to communicate with the gateway. During the uplink communication time slot of a control loop, the sensor sends the state information to the gateway. During the downlink communication time slot of a control loop, the actuator of that control loop listens to a control command from the controller forwarded by the gateway.

In RTPL, all nodes (sensors and actuators) operate in time slots. Since a transmission on the spreading factor 7 with 125 kHz bandwidth results in the smallest transmission time, we calculate the length of a time slot based on 10 byte packet transmission (suitable packet size to comply with the maximum dwell time restriction in the US on SF10 and bandwidth of 125 kHz) and its acknowledgment using spreading SF7 and bandwidth of 125 kHz. We represent the transmission time of a 10 byte packet using spreading factor 7 and 125 kHz bandwidth by $T_{tx}(SF7, 125 \text{ kHz})$ and transmission time of acknowledgments using spreading factor 7 and 125 kHz bandwidth by $T_{ack}(SF7, 125 \text{ kHz})$. In the calculation of the time slot length, we also include channel switching time, packet decoding time, and a short interval (few ms) called guard interval to handle time synchronization errors. Considering $T_{dc,sc,g}$ represents the sum of packet decoding time at the gateway, channel switching time, and guard interval, the length of the time slot is given by (1).

$$T_k = T_{tx}(SF7, 125 \text{ kHz}) + T_{ack}(SF7, 125 \text{ kHz}) + T_{dc,sc,g} \quad (1)$$

The different components used in the computation of the time slot are illustrated in Fig. 5. An uplink transmission for a control loop happens on an UCP, while a downlink transmission is done on a DCP, and they can happen concurrently. Since the packet transmission time changes with spreading factor, the number of time slots required to transmit and acknowledge a packet also varies with UCP and DCP. The time to transmit a packet almost doubles as with an increase in SF. For example, transmission and acknowledgment of a 10 byte packet on a UCP with spreading factor 7 takes 1 time slot. However, transmission and acknowledgment of a 10 byte packet on a UCP with spreading factor 8 take 2 time slots and that on spreading factor 10 takes less than 8 time slots.

If there are m UCPs then m nodes can transmit together to the gateway. he gateway can send control commands to all m nodes through a single broadcast message. However, when $n > m$,

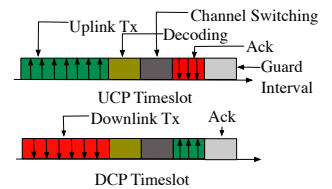


Fig. 5. Uplink and downlink time slot.

packets need to be prioritized. Unlike CPU task scheduling, wireless nodes are distributed and unaware of other nodes' priorities without which they should not transmit as the packets may collide. To handle this problem, a centralized approach can be adopted where the gateway tells which node will send next on each UCP. Such an approach will require all nodes to remain awake most of the time to listen to the gateway. This is not practical for LPWAN as it is highly energy consuming. Hence, we shall adopt an *autonomous approach* where the nodes can decide when to transmit. We enable this by giving all nodes the periods of all nodes at the beginning of network operation or if/when periods change. Since time is synchronized, a node knows when the others will have packets.

Real-time scheduling has two broad approaches – *global* and *partitioned*. In the context of LPWAN, in *global scheduling*, a control loop is dynamically assigned UCPs and priorities are global across network. In *partitioned scheduling*, the set of control loops are partitioned into m non-overlapping subsets, called *partitions*, each partition being assigned one UCP. The control loops of a partition will always use that UCP.

Global scheduling approach is inefficient for real-time communication as it can suffer from *priority violation* problem due to link unreliability and concurrent transmission in LPWAN. Specifically, when multiple nodes transmit to the gateway, a higher priority transmission may fail while the lower-priority ones succeed, raising a scenario of retransmitting a higher-priority packet after transmitting a lower priority one. To avoid such problems, we will resort to the *partitioned scheduling approach* where the priorities are only local to a partition (no global priority), and thus the priority violation problem is avoided.

6.2 Control Loop Partitioning in RTPL

We adopt the concept of partitioned scheduling in LPWAN as follows.

The gateway partitions the control loops into disjoint groups and each group is assigned a unique UCP and DCP. The uplink transmissions of the control loops happen on the assigned UCP and the downlink transmissions happen on the assigned DCP. Loops in one partition do not interfere with those in another. This greatly simplifies the real-time communication protocol and reduces overhead. For example, a control loop will need to know the periods of only the control loops of its own partition. This information only needs to be disseminated once before the start of network operation. We consider a *static* partitioning for RTPL i.e., the partitions remain unchanged once the partitioning algorithm is executed and the loops are assigned UCP/ DCP pairs. However, the partitioning algorithm may be re-run during the addition of new control loops, and the new partition information may be broadcasted. In TSCH, a time slot is assigned to a packet transmission and its acknowledgement. At the start of a new time slot, control loop l_i can preempt another control loop l_j . Hence, we adopt a *preemptive scheduling* approach, simplifying scheduling.

In each partition, packets are scheduled based on fixed priority such as *Deadline Monotonic (DM)* or dynamic priority such as *Earliest Deadline First (EDF)*. EDF schedules a task based on its absolute deadline. We break ties based on predetermined policy, such as first come first serve or lower index first, which is distributed to nodes during network deployment. Since EDF is optimal for preemptive scheduling on a single processor [25], it can offer high schedulability. Thus, we consider partitioned EDF scheduling for RTPL.

During the network deployment, each control loop is assigned a fixed number of retransmission attempts along uplink and downlink communication path to account for link failures. The gateway broadcasts the packet transmission period and the number of retransmission attempts on uplink and downlink communication paths for all control loops within the same partitions. Using the period and number of retransmissions, nodes compute a packet transmission or reception time using the EDF scheduler. Sensors compute the packet transmission time to transmit a packet,

while actuators compute packet reception time to receive control command from the gateway. If the packet transmission fails on the first time slot, a node may use its retransmission slots to successfully transmit (or receive) a packet. Upon successful transmission, nodes sleep until their next packet transmission time to save energy.

6.3 Challenges of Partitioning in RTPL

There are a number of key challenges in adopting partitioning techniques to LoRaWAN as described below.

First, multiprocessor task partitioning assumes that a task's execution requirement remains same on all processors. However, in LoRaWAN, the number of time slots required to transmit a packet varies across UCP/ DCP. Furthermore, to improve the reliability of a packet transmission, the nodes (sensors and actuators) can use additional time slots for retransmissions, similar to existing WSN technologies [18]. To ensure a high reliability of communication, the sensor of control loop ℓ_i allocates $r_i^{up}(c_k, \delta_k)$ time slots to transmit a packet on UCP c_k and receive an acknowledgment on DCP δ_k . Similarly, the actuator allocates $r_i^{down}(c_k, \delta_k)$ time slots to receive a control message of ℓ_i on DCP δ_k and transmit an acknowledgment on UCP c_k . Note that the values of $r_i^{up}(c_k, \delta_k)$ and $r_i^{down}(c_k, \delta_k)$ are pre-computed by the network manager and disseminated to all nodes during network deployment. Similar to task worst-case execution time on processor, we define *worst-case entailed time (WCET)* for a control loop ℓ_i as the total number of time slots allocated for its uplink and downlink transmissions. That is, the WCET of control loop ℓ_i , is given as $r_i(c_k, \delta_k) = r_i^{up}(c_k, \delta_k) + r_i^{down}(c_k, \delta_k)$. The WCET of a control loop represents its time of successful end-to-end (sensor to actuator) communication in the poorest conditions (requiring a retransmission in each allocated time slot).

Second, the *utilization* of a control loop, given by the ratio of WCET and its period, is used in traditional partitioning algorithms to determine the schedulability of a loop in a partition. However, in processor scheduling the WCET (standing for worst-case execution time) of a task is considered fixed and known prior to the partitioning. In RTPL, the WCET of a loop depends on its assignment to a UCP/ DCP.

Third, the position of the sensors and actuator may limit the availability of some UCPs with lower SFs to the control loop. Typically, the longer the distance between a node (sensor/actuator) and the gateway, the higher the required SF for a successful transmission. Thus, control loops may be constrained to operate on a specific set of UCPs based on the SNR at the gateway. While the bin-packing algorithms are traditionally adopted in task partitioning on multiprocessor platforms [25], directly applying those algorithms in RTPL with multiple UCPs and DCPs may lead to unschedulability even under a very low network utilization.

Finally, the number of UCPs m is usually greater than number of DCPs m' due to regional regulations. For example, in North America 64 uplink channels are available, while the number of downlink channels is 8. *Thus, the same control loop is scheduled in parts as uplink and downlink communication on different sets of UCPs and DCPs.* This is challenging because the partitioning in the uplink impacts the partitioning in the downlink and vice versa. Such unique scenario has never been explored in traditional real-time scheduling and requires a new approach. In multiprocessor scheduling, a task upon partitioning executes solely on its assigned processor.

Due to the above mentioned challenges, an optimal partitioning for RTPL can be a significantly hard problem. We do not formulate and identify the complexity class of the problem. We leave the formulation and classification as an open problem that we will study in the future. We develop a partitioning heuristic for RTPL, described in the following section.

6.4 The Partitioning Algorithm

We develop the partitioning algorithm for RTPL by addressing the challenges described in the above section and by incorporating the characteristics of a LoRaWAN network into the well-known bin-packing heuristics. We first describe the algorithm assuming the same number of UCPs and DCPs. Later, we extend it to handle cases where the total number of DCPs is less than that of UCPs.

Assuming an equal number of UCPs and DCPs, we first pair UCPs and DCPs with the same spreading factor. A control loop ℓ_i that is allocated a UCP c_k is also allocated its corresponding DCP, δ_k . The sensor of control loop ℓ_i uses UCP c_k and DCP δ_k to transmit a message and receive an acknowledgment, respectively. Similarly, the actuator of ℓ_i uses DCP δ_k to receive a control message and UCP c_k to acknowledge the reception. In such a situation, if the acknowledgment transmission by the gateway is not synchronized with the acknowledgment transmission by the actuator, it can cause packet collisions. To avoid such collision, we enforce the acknowledgments to start at the same time by (1) employing the same number of time slots ($\alpha_i(c_k, \delta_k)$) for sensors and actuators of ℓ_i to transmit sensor messages and receive control messages, respectively, and (2) forcing the gateway and actuator to start acknowledgments for ℓ_i at the same time within $\alpha_i(c_k, \delta_k)$ time slots. Typically, a sensor uses 125 kHz bandwidth to transmit a sensor message, while the gateway uses 500 kHz bandwidth to transmit a control message. Thus, the number of time slots required to transmit an uplink packet from the sensor is greater than that required to receive a downlink packet at the actuator of the same control loop. When the number of UCPs is the same as the number of DCPs, we can compute $\alpha_i(c_k, \delta_k)$ as the number of time slots allocated to the sensor of control loop ℓ_i , i.e., $\alpha_i(c_k, \delta_k) = r_i^{uP}(c_k, \delta_k)$. Using $\alpha_i(c_k, \delta_k)$, the WCET of control loop ℓ_i can be computed as $r_i(c_k, \delta_k) = 2\alpha_i(c_k, \delta_k)$.

Input: Set of all control loops L , Set of all UCP/DCP pair, V

Output: Set of Partitions $\{\pi(c_k, \delta_k) \mid k = 1, 2, \dots, m\}$

for $l_i \in L$ **do**

$V_i \leftarrow \text{ComputeAvailableCP}(l_i)$

end for

$L' \leftarrow \text{sort } L \text{ in increasing order of } |V_i|$

for $l_i \in L'$ **do**

assigned \leftarrow FALSE

for (c_k, δ_k) in V_i **do**

$\rho(c_k, \delta_k) \leftarrow 1 - u(i, k) - \sum_{\ell_j \in \pi(c_k, \delta_k)} u(j, k)$ {Compute Remaining capacity of UCP/DCP pair}

end for

$P' \leftarrow \text{sort } V_i \text{ in decreasing order of } \rho$

for (c_k, δ_k) in P' **do**

if $\sum_{\ell_j \in \pi(c_k, \delta_k)} u(j, k) \leq 1$ **then**

$\pi(c_k, \delta_k) \leftarrow l_i$

assigned \leftarrow TRUE

end if

end for

if assigned \neq TRUE **then**

return FAILURE

end if

end for

Algorithm 1: Worst-fit-UCP-increasing

In RTPL, the performance of bin-packing heuristics heavily depends on the ordering of the control loops to be assigned. One popular approach is to order the control loops based on their utilization.

However, in this case, the utilization of a control loop varies with the assignment. Specifically, the utilization of a control loop ℓ_i with period p_i that is assigned to a UCP/DCP pair c_k, δ_k is given by $u_{i,k} = \frac{r_i(c_k, \delta_k)}{p_i}$ where the WCET $r_i(c_k, \delta_k)$ of ℓ_i depends on the UCP/DCP pair (c_k, δ_k) . Hence, the actual utilization of the control loop can only be determined after an assignment. Furthermore, some control loops may be restricted to fewer usable spreading factors due to low SNR, and longer distance between node (sensor/actuator) and the gateway. To address the above limitation, we propose a partitioning heuristic called *worst-fit-UCP-increasing* for RTPL. Algorithm 1 shows the pseudocode for our partitioning heuristic. Since some SFs are not usable by a control loop due to low SNR values, the heuristic starts by computing the smallest acceptable SF for each control loop. The smallest acceptable SF for a control loop is the smallest SF that enables reliable transmission both from sensor to gateway and from gateway to actuator. Note that the smallest acceptable SF can be computed during network deployment through physical experiments or well-known radio propagation loss models [53]. The heuristic then generates the list of acceptable UCP/DCP pairs by selecting the pairs that have a spreading factor greater than or equal to the smallest acceptable spreading factor. We denote the set of acceptable UCP/DCP pairs of a control loop ℓ_i by $V_i \subseteq V$, where V is the set of all UCP/DCP pairs.

As control loops with the lowest number of acceptable UCP/DCP pairs are more likely to cause a control loop set to be unschedulable, we order the nodes in increasing order of number of acceptable UCP/DCP pairs. The algorithm iterates over the ordered list of control loops. For a control loop ℓ_i , it first computes the remaining capacity. Considering $\pi(c_k, \delta_k)$ as the set of control loops currently allocated to UCP/DCP pair (c_k, δ_k) , we define the *remaining capacity* of (c_k, δ_k) pair as the difference between 1 and sum of the utilization of ℓ_i on UCP/DCP pair (c_k, δ_k) and the total utilization of the (c_k, δ_k) , i.e., $1 - u(i, k) - \sum_{\ell_j \in \pi(c_k, \delta_k)} u(j, k)$. Note that the remaining capacity of c_k includes the utilization of ℓ_i , and a remaining capacity of 0 indicates that the allocating ℓ_i maxes out the utilization of the UCP/DCP pair (c_k, δ_k) . Similarly, a negative remaining capacity indicates that allocating the control loop ℓ_i to (c_k, δ_k) causes packet failures. Upon computing the remaining capacity, the algorithm orders the acceptable UCP/DCP pairs in the decreasing order of their remaining capacity. It iteratively selects one UCP/DCP pair and verifies if ℓ_i is schedulable on the selected pair. To test the schedulability of the control loop, we use the **RTPL schedulability test for UCP/DCP pair** given by:

$$\sum_{\ell_j \in \pi(c_k, \delta_k)} u(j, k) \leq 1 \quad (2)$$

The schedulability test for UCP/DCP pair is an exact test for a single communication path. Since a control loop schedulable on a single communication path is also said to be schedulable on two communication paths (i.e., UCP/DCP pair), the schedulability test for UCP/DCP pair is only a sufficient schedulability condition for a single partition in RTPL. If ℓ_i is schedulable, the algorithm continues with the next control loop. If a control loop is not schedulable on any acceptable UCP/DCP pairs, then it stops with a notification that the control loop set cannot be partitioned. Upon the completion of the heuristic, *if the partitioning is successful, i.e., all control loops are assigned a UCP/DCP pair, it is a guarantee that they are schedulable* (that is, each control loop shall meet its deadline). This comes from the fact that each partition is schedulable (as it passed the schedulability test). Note that a successful partitioning is only a sufficient condition for schedulability. Therefore, in case of unsuccessful partitioning, nothing can be concluded about the schedulability of the loops. In the worst-fit-UCP-increasing partitioning, the choice of assigning UCP/DCP pairs based on increasing remaining utilization stems from the fact that it balances the number of control loops

assigned across all available UCPs. Thus, it minimizes the average energy consumption of all nodes in the network.

Upon partitioning, every node is informed of its UCP/ DCP pair and the periods of other control loops on its partition. Nodes (sensors, gateway and actuators) can locally generate a schedule of all transmission based on the EDF policy. This schedule can be used to sleep, transmit/receive packets, and transmit/receive acknowledgments. Note that our proposed partitioning approach can be generalized to any number of UCP/ DCP pairs and can be extended to SigFox which is similar to LoRaWAN. Furthermore, our approach can be extended to any of the following bin-packing heuristics [34], which we call as heuristic variants of RTPL.

RTPL-BFD: The *best-fit-decreasing* heuristic places each item in the bin where it leaves the smallest empty space. RTPL-BFD is a variation of RTPL following the best-fit decreasing heuristic, where the nodes are orde in decreasing order of utilization and the UCP with the least remaining capacity is chosen first for assigning a loop.

RTPL-WFD: The *worst-fit-decreasing* heuristic places each item in the bin where it leaves the largest empty space, creating a new bin if necessary. RTPL-WFD is a variation of RTPL following the worst-fit decreasing heuristic, where loops are ordered in decreasing order of utilization.

RTPL-FFD: The *first-fit-decreasing* heuristic places each item in the first bin where it fits, moving sequentially through bins. RTPL-FFD is a variation of RTPL following the firs-fit decreasing heuristic, where the loop is assigned to the first UCP with enough capacity to schedule it. Loops are ordered in decreasing order of utilization.

RTPL-BFUI: The RTPL-best-fit-UCP-increasing heuristic is a variation of RTPL based on best-fit heuristic, where the nodes are ordered by available UCPs.

RTPL-FFUI: The RTPL-first-fit-UCP-increasing heuristic is a variation of RTPL based on first-fit heuristic, where the nodes are ordered by available UCPs.

Schedulability Analysis:

In real-time process control applications, schedulability analysis is widely used to determine, both during the design time and for online admission control, if a given set of control loops/flows can meet their deadlines. It determines whether the end-to-end communication between a sensor and an actuator could be completed within the deadline. Hence, it facilitates proactive network management by enabling network manager to plan ahead and adjust workloads.

If there exists a transmission schedule ensuring no deadlines are missed, the set of control loops is termed as *schedulable*. The two main approaches for schedulability analysis are *end-to-end delay bound analysis* and *utilization bound analaysis*. By specifying the maximum possible utilization of all flows in the network, a utilization bound analysis determines whether the flows are schedulable based on whether the total utilization remains within the network's maximum achievable utilization. The utilization-based schedulability test is regarded as one of the most efficient and effective tests due to its notably low run time overhead. A schedulability test is *sufficient* if any set of control loops, determined as schedulable by a schedulability test, is indeed schedulable.

Upon the completion of our proposed partitioning heuristic, if the partitioning is successful, it is a guarantee that each control loop is schedulable. This comes from the fact that each partition is schedulable (as it passed the schedulability test). However, running the complete partitioning algorithm to check schedulability is overly time consuming and computationally complex. To reduce the computation complexity, we propose a ***sufficient condition for partitionability***. If this condition is fulfilled, successful partitioning of control loops is ensured, thereby ultimately guaranteeing the schedulability of the control loops.

Many task partitioning algorithms for CPU scheduling were studied in the literature based on various bin packing heuristics. The first-fit-decreasing bin-packing heuristic provides an effective partitioning in real-time task scheduling on multiprocessor [21]. It assigns a task to the first

processor with enough remaining capacity, measured by considering each of its already assigned task's *demand bound function* (DBF) [22]– the largest cumulative execution time requirement of all jobs (packets of a control loop in our case) generated by the task (control loop) with both release times and deadlines within a contiguous time interval, t . Considering r_i as the worst case entailed time (WCET) needed by a control loop and **utilization** $\mu_i = \frac{r_i}{p_i}$, DBF and its approximation [20] can be extended to a control loop ℓ_i with period p_i and deadline d_i as follows

$$\begin{aligned} \text{DBF}(\ell_i, t) &= \max \left(0, \left(\left\lfloor \frac{t - d_i}{p_i} \right\rfloor + 1 \right) r_i \right) \\ &\approx \begin{cases} 0, & \text{if } t < d_i; \\ r_i + \mu_i(t - d_i), & \text{otherwise.} \end{cases} \end{aligned} \quad (3)$$

Based upon the DBF function, the *load* of the task system, denoted by λ , is defined as

$$\lambda = \max_{t>0} \left(\frac{\sum_{i=1}^n \text{DBF}(\ell_i, t)}{t} \right) \quad (4)$$

The schedulability analysis in [21] provides a sufficient schedulability condition in terms of λ , $\mu_{\max} = \max\{\mu_i | 1 \leq i \leq n\}$, and $\mu_{\text{sum}} = \sum_{i=1}^n \mu_i$, given by

$$m \geq \frac{2(\lambda - \mu_{\max}) + \mu_{\text{sum}}}{1 - \mu_{\max}}. \quad (5)$$

If the condition is met, all tasks meet deadlines upon partitioning. The above schedulability condition was proposed for any ordering, hence we may tailor it with RTPL partitioning algorithm to derive the sufficient condition for partitionability as described below. We consider a network to processor scheduling mapping, where (i) a unique UCP is analogous to a processor and (ii) a packet transmission and its associated ACK is analogous to a task execution on a processor for one time unit. RTPL partitioning algorithm assigns the control loops in non-increasing order of the size of the subset of available UCPs as ℓ_1, ℓ_2, \dots , where $|V_i| \leq |V_{i+1}|$ or $|V_i| = |V_{i+1}|$ and $u_{i,k} \geq u_{i+1,k}$ for $1 \leq i \leq n$. As there can be multiple nodes with the same number of UCPs, we break ties by ordering them in non-increasing order of their utilization in the UCP $\pi_k \in V_i$ where $\Phi(p_{i_k}) = \gamma_i$. We allocate 1 time slot for the uplink transmission and 1 time slot for the downlink transmission, without considering the retransmissions, and get the WCET, $r_i = 2$. Note that in LoRa, the number of time slots requi to transmit a packet varies across UCP/ DCP. Thus, we consider time slot lengths of UCPs and DCPs to be based on their spreading factor. Particularly, we consider the longest time slot across all the spreading factors. Leveraging the above definitions of r_i and μ_i , we rewrite the Eq. (3) as follows.

$$\begin{aligned} \text{DBF}'(\ell_i, t) &= \max \left(0, \left(\left\lfloor \frac{t - d_i}{p_i} \right\rfloor + 1 \right) 2 \right) \\ &\approx \begin{cases} 0, & \text{if } t < d_i; \\ 2 + \frac{2(t - d_i)}{p_i}, & \text{otherwise.} \end{cases} \end{aligned} \quad (6)$$

Using Eq.(6), we can revise Eq.(4) as follows.

$$\lambda' = \max_{t>0} \left(\frac{\sum_{i=1}^n \text{DBF}'(\ell_i, t)}{t} \right) \quad (7)$$

Finally, using Eq. (7), we revise Eq. (5) to give the sufficient condition for partitionability, where a control loop ℓ_i can be successfully scheduled on m UCPs under EDF scheduling as.

$$m \geq \frac{2(\lambda' - \mu_{\max}) + \mu_{\text{sum}}}{1 - \mu_{\max}} \quad (8)$$

Note that since we use the longest time slot to derive the above condition for RTPL, this condition is analogous to identical processors and is slightly pessimistic.

6.5 Handling Retransmissions

Due to channel noise, a control loop may face transmission failures. Thus, we pre-assign time slots for possible retransmissions. These slots may remain unused if the transmission succeeds in the first slot. Consequently, the partitioning result will also be affected. This is especially challenging for the UCPs and DCPs which use the higher SFs as their capacity is low due to the high time slot length. Note that, the UCP and DCPs using a higher SFs are robust to interference and rarely trigger retransmissions. Thus, instead of using a fixed number retransmission slot for all partitions, we propose to use a variable number of retransmissions depending on the UCP/ DCP pair.

Specifically, we assign $\kappa(c_k, \delta_k)$ slots for each transmission made on UCP/ DCP pair (c_k, δ_k) . A value of $\kappa(c_k, \delta_k)$ can be considered as an upper bound of expected number of transmissions on UCP/ DCP pair (c_k, δ_k) and can be predetermined from historical data. Thus, the WCET for each control loop assigned to UCP/ DCP pair becomes $r_i(c_k, \delta_k) = 2\alpha_i(c_k, \delta_k)\kappa(c_k, \delta_k)$. Note that, The same number retransmissions $\kappa(c_k, \delta_k)$ is used for all UCP c_k with same SF. This maintains the validity of our downlink partition which groups the control messages for loops assigned to the UCPs with same SF and transmits them using the same number of retransmissions $\kappa(c_k, \delta_k)$.

6.6 Handling Workload and Network Dynamics

Due to the dynamic nature of network workloads, the period of a control loop may change over time to meet application requirements. This may lead to increased utilization of the control loop. The partitioning heuristic for RTPL typically balances the remaining capacity across all UCP/ DCP pairs. This means that small variations of workload may be handled without re-running the partitioning heuristic, as long as the current partition is still schedulable under the new utilization of the loop. Thus, in case of small variations in workload, the network server can notify about it to the other loops in the partition, which can then update their schedules accordingly, without needing to re-execute the partitioning heuristic. However, the partitioning heuristic is re-executed if the workload changes significantly. Similarly, if the signal-to-noise (SNR) ratio on a channel becomes extremely low, i.e, a channel becomes overly noisy, the gateway can blacklist it and re-run the partitioning algorithm excluding that channel.

6.7 Handling Different Numbers of UCPs and DCPs

Regional regulations typically limit the number of DCPs to be less than the number of UCPs, e.g., in the US, number of UCPs is 64 while the number of DCPs is 8. This is because DCPs require separate radios, due to which a high number of DCPs become expensive. One approach to address this limitation is to use only m' UCPs in the uplink partition, but this can lead to under-utilization of the network.

LoRaWAN uses a narrow bandwidth of 125 kHz for uplink communication and a broad bandwidth of 500 kHz for downlink communication. The higher bandwidth of downlink communication allows us to group packets from multiple UCPs on a single DCP. In RTPL, we group control messages from multiple control loops into a single downlink communication message. This grouping of multiple packets changes the mapping between UCPs and DCPs to many-to-one, where multiple UCPs with

the same SF are mapped to a single DCP. Specifically, we generate a partition $\pi(c_i, c_j, \dots, c_k, \delta_p)$ where $\Phi(c_i) = \Phi(c_j) = \dots = \Phi(c_k) = \Phi(\delta_p)$. For example, c_1, c_2, c_3 and c_4 can be mapped to δ_1 . In such a mapping, if the gateway receives a packet from control loops ℓ_1, ℓ_2, ℓ_3 and ℓ_4 on UCPs c_1, c_2, c_3 and c_4 , respectively, then the gateway acknowledges all transmission with a single grouped message on δ_1 . Similarly, the gateway also groups the control messages of ℓ_1, ℓ_2, ℓ_3 and ℓ_4 and transmits the grouped message on δ_1 .

Packet grouping results in increase of the number of time slots required by the gateway. However, we have observed that such overhead is minimal for LoRaWAN since it uses non-uniform bandwidths for uplink and downlink communications, with the downlink bandwidth being higher. This results in a 74% reduction in airtime for downlink communication, which offsets the increase in time slots caused by packet grouping. For example, transmission of a 10 byte packet on spreading factor 10 with a 125 kHz bandwidth takes 320.7 ms while it takes only 92.7 ms on 500 kHz bandwidth [3]. Thus, LoRaWAN enables the grouping of 4 downlink packets with minimal overhead. Note that the number of time slots allocated to the sensor may vary from that allocated to the actuator of the same control loop actuator depending on the grouping used. When the number of UCPs is greater than the number of DCPs, we compute the WCET of a control loop ℓ_i based on the maximum of number of time slots allocated to a sensor of ℓ_i and number of time slots allocated to an actuator of ℓ_i . Specifically, WCET of control loop ℓ_i can be computed as $r_i(c_i, c_j, \dots, c_k, \delta_p) = 2\alpha_i(c_i, c_j, \dots, c_k, \delta_p) = 2 \max(r_i^{up}(c_i, c_j, \dots, c_k, \delta_p), r_i^{down}(c_i, c_j, \dots, c_k, \delta_p))$. The partitions can now be generated using the new grouping and WCET. Note that RTPL is a general protocol, with an extended capability to handle channel asymmetry without throttling its potential for massive concurrent reception. It can work effectively when the number of available DCPs is same as the number of UCPs, as well as when the total number of UCPs are greater than the total number of DCPs.

6.8 Handling Channel Hopping and Duty-Cycling

There is a maximum dwell time regulation of 400ms on any channel for LoRa nodes in the US. However, nodes with an output power of 21dBm and using only 8 uplink channels are exempt from this channel hopping regulation [3]. The nodes are further required to implement pseudo-random channel hopping before every transmission in case a node utilizes at least 50 uplink channels. To handle the channel hopping regulation, we can extend our framework by assigning each UCP a predetermined sequence of frequency hopping. The gateway shares this sequence with the nodes in the partition. After each time slot, the node hops to the next channel as per the predetermined sequence. Although the physical channel of a UCP changes due to hopping, the partitioning remains unchanged because all nodes within a partition switch to the same channel.

In some regions, such as North America, LoRaWAN networks can host a relatively high workload and applications with real-time requirements, as there are no duty-cycle regulations. While in some regions there are tight duty cycle requirements. For example, to ensure fair usage of the spectrum in Europe, the duty cycle is typically set to 1% when using the ALOHA protocol and no restrictions when using the LBT protocol. ETSI (European Telecommunications Standards Institute) has segmented the European ISM frequency band into 6 sub-bands (K, L, M, N, P, and Q) and imposed additional restrictions on the duty cycle as (K 0.1%, L 1%, M 1%, N 0.1%, P 10% and Q 1%) [6]. RTPL can be easily extended to handle duty-cycling requirements in such regions. This can be done by changing the partitioning approach by stalling UCP/DCP pairs with dummy control loops. Specifically, for each UCP/DCP pair with a duty-cycle, d , we assign a dummy control loop with a utilization of $(1 - d)$. This ensures that our partitioning approach complies with the duty-cycle regulation. It is understandable that with duty-cycle requirement the network may support lesser workload.

6.9 Summarizing RTPL

In this section, we summarize RTPL. In RTPL, node partitioning and UCP/DCP pair assignment occur at the gateway, while packet transmission scheduling and retransmissions are managed at the nodes.

In RTPL, the gateway first pairs UCPs and DCPs with the same spreading factor to create partitions. If the number of available UCPs is the same as that of available DCPs, a partition consists of one UCP and DCP with the same SF, represented as $\pi(c_i, \delta_j)$. If the number of UCPs is greater than the number of DCPs, a partition contains multiple UCPs and one DCP, wherein all UCPs and DCPs have the same SF, represented as $\pi(c_i, c_j, \dots, c_k, \delta_p)$. The set of all partitions is represented as Π .

Upon calculating the partitions, the gateway calculates the time slot length using Eq. (1). When the number of UCPs is greater than the number of DCPs, the gateway groups the control command messages of multiple control loops into a single packet. The number of control commands grouped in a single packet is upper bounded by the number of UCPs in a single partition. The time slot length is calculated using the transmission time for transmitting a single grouped packet on a DCP with the smallest SF, acknowledgment from one actuator, and $T_{dc,sc,g}$.

The gateway collects SNR information, the distance of the nodes from the gateway, and the smallest acceptable SF for each control loop ℓ_i . The smallest acceptable SF is used to identify the set of acceptable partitions for each control ℓ_i . The set of acceptable partitions of a control loop ℓ_i is denoted by $\Pi_i \subseteq \Pi$. The gateway also computes the number of retransmission attempts for each control loop ℓ_i to meet the reliability requirement and its WCET. The WCET of a control loop ℓ_i is also calculated as $r_i(c_i, c_j, \dots, c_k, \delta_p) = 2\alpha_i(c_i, c_j, \dots, c_k, \delta_p) = 2 \max(r_i^{up}(c_i, c_j, \dots, c_k, \delta_p), r_i^{down}(c_i, c_j, \dots, c_k, \delta_p))$.

Upon calculating the WCET of each control loop, the gateway assigns nodes to partitions using the worst-fit-UCP-partitioning algorithm, as described in Sec. 6.4. The algorithm iteratively selects a partition from the set of acceptable partitions and verifies if a control loop is schedulable on the selected partition using the RTPL schedulability test given by Eq. (2). When a control loop is schedulable on a selected partition, it is assigned to the selected partition, and the algorithm continues with the following control loop. If a control loop is not schedulable on any acceptable partition, the algorithm stops with a notification that the control loop is not schedulable. RTPL algorithm running at the gateway is presented in Algorithm 2.

Input : Set of control loops L , set of all available UCPs U , and set of all available DCPs D

Output : Set of partitions Π

STEP 1: Generate the set of partitions Π .

STEP 2: Compute the time slot length for a partition with the smallest SF.

STEP 3: Repeat Steps 4 - 5 for each control loop ℓ_i .

STEP 4: Compute the smallest_SF for loop ℓ_i based on the SNR and distance to gateway.

STEP 5: Compute the set of acceptable partitions Π_i for control loop ℓ_i .

STEP 6: Partition the control loops using Algorithm 1, worst-fit-UCP-increasing partitioning algorithm.

STEP 7: Repeat Step 8 for each partition.

STEP 8: Disseminate the periods and WCET of all control loops in a partition to all end devices of the partition.

Algorithm 2: RTPL at the gateway.

The RTPL schedulability for a partition with one UCP and DCP is the same as the schedulability condition for a partition with multiple UCPs and one DCP. When the number of UCPs is greater than

the number of DCPs, one downlink channel simulates the behavior of multiple downlink channels by transmitting multiple downlink communication in a single time slot. Specifically, the gateway acknowledges multiple uplink transmissions on one DCP within the same slot. Additionally, the gateway transmits multiple control commands on one DCP in one slot. Thus, RTPL schedulability for a partition with one UCP and DCP is the same as the schedulability condition for a partition with multiple UCPs and one DCP.

Upon receiving the periods and WCET of all control loops within a partition, nodes generate and follow the EDF schedule locally. Specifically, sensors generate the EDF schedule to identify the next packet transmission slot. If the packet transmission is successful on the first attempt, the sensor sleeps until it is time to transmit the next packet. However, if the packet transmission fails, the sensor continually finds the next retransmission slot from the EDF schedule and attempts to retransmit the packet until it succeeds or runs out of retransmission slots. Similarly, an actuator generates an EDF schedule to identify the next slot to receive a packet from the gateway. Upon receiving a packet, actuator sleeps until its next scheduled packet reception slot.

7 IMPLEMENTATION

We have implemented RTPL using Raspberry Pi LoRa HAT (LoRaHAT) and Dragino LG308 LoRaWAN gateway (GW) as LoRa nodes and gateway, respectively [4]. LoRaHAT is a commercial-of-the-shelf (COTS) module that offers low-cost, ultra low-power LoRaWAN implementation. Its design is based on the SEMTECH SX1276 LoRa transceiver [11]. The GW implements SEMTECH packet forwarder and is compatible with the existing LoRaWAN protocol. It includes one SX1301 chip, which contains a LoRa IP concentrator, and two SX1257 chips allowing for ten parallel demodulation paths. We use the Things Stack network server from The Things Industries [16]. It is an open source and widely used LoRaWAN network server. Due to the lack of a working implementation of LoRaWAN class B mode at the time of experiment, we emulate class B operation with the help of Internet-based time synchronization. Namely, all Raspberry pi's were connected to the Internet through WiFi, thus even without a dedicated time-synchronization protocol they all had almost similar timestamps. We use a guard band of 100 ms to reduce any packet collisions due to time synchronization errors and to account for the clock drifts over a long synchronization period. In our experiments, we store the EDF schedule locally at each node. We assigned a time slot of 7 seconds based on Eq. (1). Note that this includes a 5 second receive window delay recommended by the The Things Stack for sending an acknowledgment.

8 EXPERIMENTS

In this section, we evaluate the performance of RTPL in different environments (indoor and outdoor) in a large metropolitan city in the US. As discussed in Section 4, there are some existing works that explore the potential of LoRa/ LoRaWAN in real-time communication and control [32, 35, 65]. However, these works are not comparable with RTPL. Specifically, the work in [65] discusses the potential of LoRaWAN to enable real-time control but does not propose any real-time protocol. The work in [35] does not provide a slot-scheduling algorithm to meet real-time requirements. The scheduling proposed in [32, 58] does not consider the different time requirements for transmissions on different spreading factors and rely on the assumption that LoRa nodes are capable of direct peer-to-peer communication. Hence, we evaluate the performance of RTPL by comparing it with LoRaWAN.

8.1 Indoor Deployment

8.1.1 Setup. The indoor experiments are carried inside a building at our location. We use the same deployment (keeping the positions of the nodes and the GW unchanged) for experiments with

LoRaWAN and RTPL. In all the experiments, we run the partitioning algorithm for RTPL. We use the following default parameters setting for experiments in this Section unless stated otherwise.

- Period: 7, 14, 56, 112 (seconds)
- SF: 7, 8, 9, 10
- Deadline = period
- Frequencies: 903.9-904.7 MHz
- Packet size: 10 bytes
- Bandwidth: Uplink: 125 kHz, Downlink: 500 kHz
- TX power: 15 dBm
- Distance: Indoor: 2 m - 6 m

Note that the performance of our algorithm only increases with shorter packet size. Thus, in our experiments, we used the maximum allowable packet size for SF10 (10 bytes) in the US as the size of our packet which shows a lower bound of our performance. We used up to 4 control loops and 4 SFs (7-10) in our experiment. As the worst-fit UCP increasing algorithm tries to balance the control loops across different assignments, each node was assigned to a different SF. For LoRaWAN, the SFs were assigned based on the adaptive data rate (ADR) algorithm.

8.1.2 Real-Time Performance.

To observe the performance of RTPL in terms of schedulability, we conduct three different experiments. We observe the schedulability of RTPL in two cases: using harmonic period and non-harmonic period. For the non-harmonic and non-harmonic period experiments, we use four control loops and observe the number of schedulable control loops. Fig. 6 shows the results of our schedulability experiments. In Fig. 6(a), all control loops are schedulable under RTPL, while one control loop is schedulable under LoRaWAN. This is due to LoRaWAN ADR mechanism that controls several parameters (e.g., SF). Fig. 6(b) shows that 100% of the control loops are schedulable with RTPL using a non-harmonic period. This is due to RTPL partitioning, which guarantees each packet meets its deadline. For LoRaWAN, 100% of the control loops are unschedulable.

8.1.3 Latency.

In this Section, we measure the end-to-end latency of the control loop of RTPL and LoRaWAN under default setting. We calculate the average and maximum latency per control loop for the duration of the hyper-period. A node generates a new packet periodically, with a period of 7s. Note that we

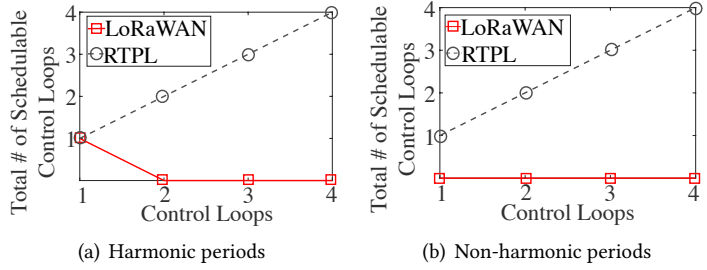


Fig. 6. Schedulability of RTPL(Indoor)

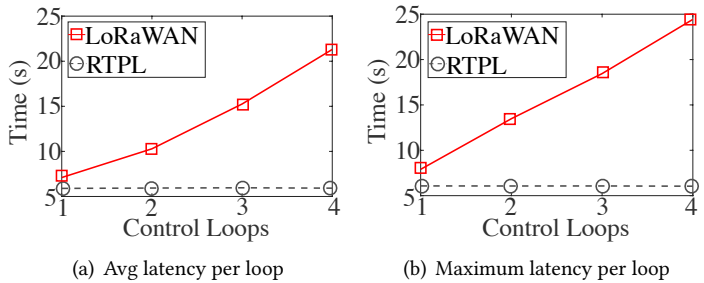


Fig. 7. Latency vs. # of control loops (Indoor)

calculate latency for packets which were received by the gateway and the corresponding downlink message was received by the actuator. In our calculation, we include the beacon time on air (approximately 160 ms) and RX delay (5 s). Fig. 7 shows the results of our experiments.

In Fig. 7(a), the average end-to-end latency for RTPL for all control loops is around 6 seconds. For LoRaWAN, the average latency for one control loop is around 7.1 seconds and it increases linearly with the number of control loops, as shown in Fig. 7(b). For four control loops, LoRaWAN average latency is 23 s. This is because packets were initially transmitted on the highest SF to ensure maximum communication range and robustness to interference, and later changed by the ADR mechanism which is recommended by Semtech [7]. The acknowledgment mechanism in LoRaWAN (the GW acknowledges uplink transmissions sequentially) also contributed to the delay.

8.1.4 Energy and Throughput. In this section, we measure the end-to-end energy consumption of the control loop and throughput of RTPL.

For the energy consumption experiment, we use the same setup as the latency experiments. To estimate the energy, we use SEMTECH SX1278 chip energy model (3.3 v voltage, TX 87 mA, RX 12 mA). We measure the average energy consumption per control loop for both RTPL and LoRaWAN. The results in Fig. 8 shows that the average energy consumption for RTPL is around 69.9 mJ and 73.1 mJ for one and four control loops, respectively. The average energy consumption for LoRaWAN is 116 mJ and 157 mJ for one and four control loops, respectively. This result show that RTPL offers energy-efficiency while guaranteeing schedulability.

To compare the throughput of RTPL and LoRaWAN, we need to measure the throughput of the control loop without considering its schedulability. For example, if a packet missed its deadline and is successfully received by both the GW and the actuator, we consider it as part of our throughput calculation. We define the throughput as the bits received by the GW (uplink) and the bits received by the actuator (downlink) for three hyperperiods (5.6 minutes) Fig. 9 shows the throughput for both LoRaWAN and RTPL. For LoRaWAN, the throughput is 4272 bps compa to 4176 bps for RTPL. The throughput for RTPL is comparable to LoRaWAN as in our setup, we consider non-schedulable control loops. This result shows that while RTPL guarantees schedulability, it does not compromise the throughput.

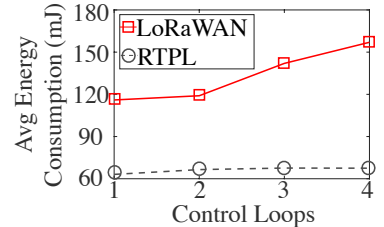


Fig. 8. Avg energy consumption (Indoor)

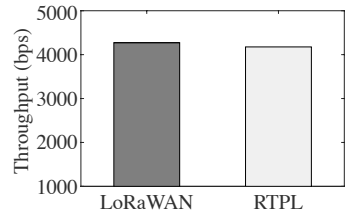


Fig. 9. Throughput (Indoor)

8.2 Outdoor Deployment

8.2.1 Setup. For the outdoor deployment, we observe the performance of RTPL at varying distances and channel conditions. Fig. 10 shows the locations and distances of the nodes and GW in a large metropolitan city in the US. The GW is placed inside a building while the nodes are placed in the locations shown on the map. In this section, we use the following default parameters settings unless stated otherwise.

- Period: 7, 14, 56, 112 (seconds)
- SF: 7, 8, 9, 10
- Deadline = period
- Frequencies: 903.9-904.7 MHz
- Packet size: 10 bytes
- Bandwidth: Uplink: 125 kHz, Downlink: 500 kHz
- TX power: 15 dBm
- Distance: Outdoor: 200 m - 500 m

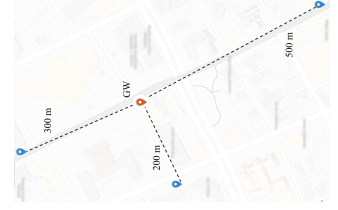


Fig. 10. Outdoor GW and nodes locations

8.2.2 Schedulability Performance. To observe the impact of distance on the schedulability performance of RTPL, we repeat the schedulability experiments for the harmonic period and non-harmonic period outdoors.

The GW is placed inside a building at our location and the nodes are placed outdoor at various locations as shown in Fig. 10. Fig. 11 shows the results of our schedulability experiments. In Fig. 11(a), 100% of the control loops are schedulable under RTPL.

Under LoRaWAN, all control loops are non-schedulable. This is possible due to the variation of channel conditions and LoRaWAN ADR mechanism resulting in increasing packet collision. Fig. 11(b) shows that 100% of the control loops are schedulable with RTPL using a non-harmonic period. For LoRaWAN, 100% of the control loops are non-schedulable.

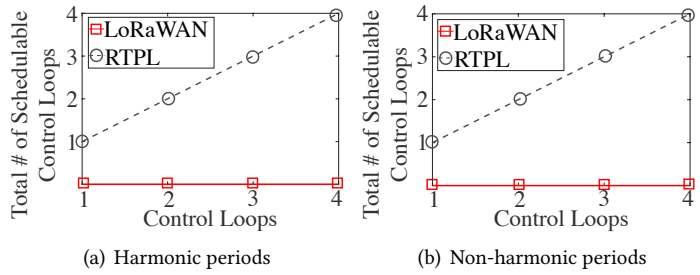


Fig. 11. Schedulability of RTPL (Outdoor)

8.2.3 Latency. We estimate the end-to-end latency of the control loop in outdoor deployment. We also calculate the average and maximum latency per control loop for the duration of the hyper-period. A node generates a new packet periodically, with a period of 7s.

Fig. 12 shows the results of the latency experiments. In Fig. 12(a), the average latency of RTPL is around 6.2 seconds for all the control loops. While the average latency for one control loop is around 7.3 seconds for LoRaWAN. With four control loops, the average latency for LoRaWAN

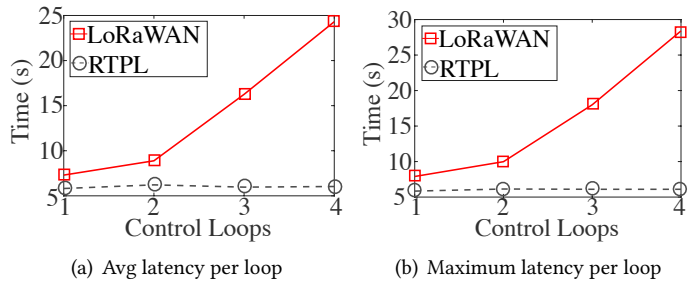


Fig. 12. Latency vs. # of control loops (Outdoor)

is around 24 seconds. In Fig. 12(b), the maximum latency for RTPL for all control loops is around 6.3 seconds. On the other hand, the maximum latency of LoRaWAN is 7.9 seconds and 28 seconds for one and four control loops, respectively.

We measure the average energy consumption per control loop for the outdoor scenario where each node generates a new packet periodically, with a period of 7s. For such low periods, we observe significantly high packet collisions. As the number of nodes are increased, we observe a significant drop in the packet reception rate of nodes. Specifically, for the setup with 3 and 4 nodes,

the first few packets were received successfully with 3-4 retransmission attempts, and the rest of the packets were not received. For the setup with 2 nodes, most of the packets were received but 4-5 retransmission attempts.

8.2.4 Energy and Throughput. In Fig. 13, the average energy consumed by a node was computed for successfully received packets. Thus, the setup with 2 nodes, needed higher energy than the setup with 3 or 4 nodes. The result shows that the average energy consumption for RTPL is around 69.9 mJ for all the control loops. The average energy consumption for LoRaWAN is 217 mJ. This implies that distance has an insignificant impact on the energy consumption of the control loop. Also, RTPL provides schedulability guarantees while remaining energy-efficient.

Next, we compare the throughput of RTPL and LoRaWAN for 5.6 minutes. For RTPL and LoRaWAN, the throughput is 4160 bps and 3512 bps, as shown in Fig. 14. This result shows that RTPL achieves better throughput in outdoor deployments than LoRaWAN.

9 SIMULATION

Here, we evaluate RTPL with extensive simulations in NS-3 with a single gateway and up to 1200 control loops. We report the schedulability ratio and total transmission energy consumption results for various setups. Our simulation model is publicly available on GitHub [8].

9.1 Setup

In our simulation, we employed LoRaWAN NS-3 module proposed in [38]. The NS-3 module offers classes and channel model designed for simulating modulation and medium access technology of LoRaWAN network, supports simulations featuring a large number of devices and offers easy monitoring of network performance. We used up to 1200 control loops with each control loop having one sensors and one actuators. The sensors and the actuators of the control loops were randomly positioned across a disc centered at the gateway, with radius up to 7000 meters. Each control loop was assigned a randomly selected period from a range of $[2^{12}, 2^{25}]$ ms. We used up to 64 UCPs and 18 DCPs and SFs 7-10. We do not use spreading factors greater than 10 for US dwell time regulation. We map 4 UCPs to a single DCP. For example, for 24 UCPs, we used 6 DCPs. We set the maximum number of retransmissions to 2 and used a timeslot length of 120 ms.

Benchmark: We compare our approach Worst-fit-UCP-increasing (RTPL-WFUI) and the described variations of RTPL i.e. RTPL-BFD, RTPL-WFD, RTPL-FFD, RTPL-BFUI and RTPL-FFUI with regular LoRaWAN.

Metric: The *schedulability ratio* is defined as the ratio of the number of control loop sets that are schedulable to the total number of control loop sets. In the first simulation, we report the schedulability ratio based on whether the partitioning was successful or not. We refer to it as *schedulability-ratio-partitioning*. In the second simulation, we report the schedulability ratio based on the schedulability analysis. In this simulation, only the sufficient condition proposed in Eq.(8) needs to be checked to evaluate schedulability. We refer to it as *schedulability-ratio-analysis*. In

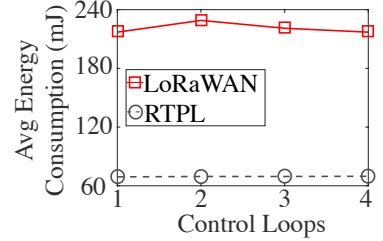


Fig. 13. Avg energy consumption (Outdoor)

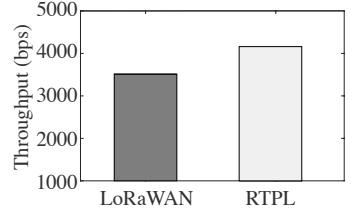


Fig. 14. Throughput (Outdoor)

addition, we also report the total transmission energy consumption for the schedulable control loop sets in both cases. We generated 30 control loop sets for each simulation and each result is averaged over 30 simulation runs of 1 hour.

9.2 Results under Varying Number of Control Loops

In this simulation, We fix the number of UCPs to 24 and the network radius to 4000 m. For each point, we generated 30 control loop sets and vary the number of control loops within a set from 200 to 1200. We report the schedulability ratio and total transmission energy consumption. In Fig. 15, we analyze the schedulability-ratio-partitioning and present the schedulability ratio and transmission energy consumption under varying number of control loops within a set for all the variants of RTPL.

In Fig. 15(a), all the control loop sets are schedulable under all variants of RTPL, except RTPL-BFD. The schedulability of the RTPL-BFD starts decreasing with an increase in number of control loops within a set. The schedulability ratio drops to 0.84 when there are 600 control loops in a set. This means that only 84% of the total control loop sets are schedulable. The schedulability ratio rapidly drops to 0.22 when the control loops in a set are further increased to 800. That implies only 22% of the control loops sets are schedulable. The schedulability ratio for RTPL-BFD drops to 0 beyond this point. This happens due to the intrinsic nature of RTPL-BFD to assign the control loop with the highest utilization to the UCP with the least remaining capacity. Thus, RTPL-BFD allocates a loop to the UCP where the WCET of that loop becomes highest. As the number of loops in This, combined with the fact that traditional best-fit decreasing heuristic does not take into account the number of available UCP/DCP pairs during the loop's assignment to a partition, leads to poor schedulability performance of RTPL-BFD. Furthermore, due to high number of collisions in traditional LoRaWAN, none of the control loop sets are schedulable in traditional LoRaWAN. The ability of RTPL to employ any variant of the bin-packing heuristics, enhances the adaptability and significantly boosts the scalability of RTPL. In fact, RTPL-WFUI, RTPL-WFD, RTPL-FFD, RTPL-BFUI, and RTPL-FFUI perform better than regular LoRaWAN with 100% schedulability.

In Fig. 15(b), we show the total transmission energy consumption for the schedulable control loop sets under the same setup. Note that, we do not show the energy consumption for traditional LoRaWAN and RTPL-BFD since these approaches are not schedulable for all sets. In this figure, as the number of control loops in a set increase, the energy consumption for all approaches steadily increase. However, RTPL-BFUI has the highest energy consumption among all the variants as it tries to assign the loops to the UCPs where the WCET of loops is highest. RTPL-FFD has the least energy consumption. On average, it has 45% less energy consumption than RTPL-BFUI, showing the energy-efficiency of our partitioning approach.

In Fig. 15(b), we show the total transmission energy consumption for the schedulable control loop sets under the same setup. Note that, we do not show the energy consumption for traditional LoRaWAN and RTPL-BFD since these approaches are not schedulable for all sets. In this figure, as the number of control loops in a set increase, the energy consumption for all approaches steadily increase. However, RTPL-BFUI has the highest energy consumption among all the variants as it tries to assign the loops to the UCPs where the WCET of loops is highest. RTPL-FFD has the least energy consumption. On average, it has 45% less energy consumption than RTPL-BFUI, showing the energy-efficiency of our partitioning approach.

9.3 Results under Varying Network Radius

In this simulation, we vary the network radius from 1000 m to 7000 m. We set the number of control loops within a set to 600 and the number of UCPs to 24. We show the result for schedulability-ratio-partitioning in Fig. 16. In Fig. 16(a), we see a sharp decline in schedulability for all partitioning

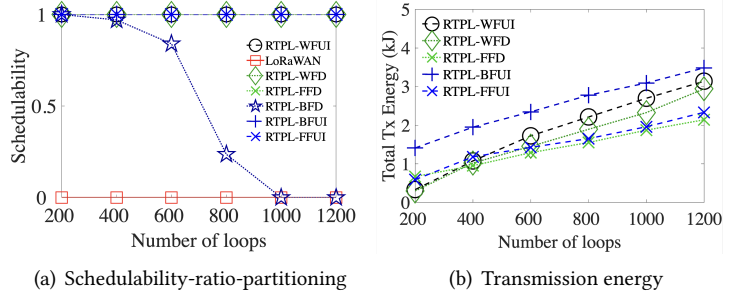


Fig. 15. Results under varying number of control loops.

approaches as the network radius increases to 5000 m. This is due to the limited number of available UCP/DCP pairs at such a large network radius. The control loops are required to use higher SFs at large radius to ensure a successful packet reception at the gateway, leading to a limited number of available UCP/DCP pairs. Therefore, the number of unschedulable cases increase at higher network radius. To establish this observation further, we set the number of UCPs to maximum allowable UCPs for LoRaWAN i.e. 64 and used the number of control loops within a set to 200. Even with such configuration, we observed that none of control loop sets were schedulable under any of the heuristics at 7000 m radius. In conclusion, all variants of RTPL and its variants show on average 30% better schedulability than regular LoRaWAN, but the better performance is limited to a radius of 5000 m only.

In Fig. 16(b), we observe that RTPL-BFUI has the highest energy consumption while RTPL-FFUI has the lowest. We observe almost constant total transmission energy consumption for all partitioning approaches till a network radius of 3000 m. However, as the radius increases, we observe a sharp increase in energy consumption for almost all approaches. This is because the nodes use higher SFs at larger network radius. However, the energy consumption of RTPL-BFUI remains constant at all radii as it by default tries to assign loops to UCPs with the highest available SF, hence requiring high transmission energy. Note that RTPL-FFUI consistently has the lowest energy consumption. In fact, on an average RTPL-FFUI performs 58% better than RTPL-BFUI.

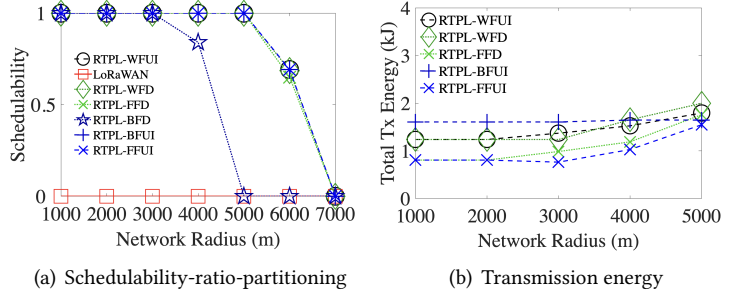


Fig. 16. Results under varying network radius.

9.4 Results under Varying Number of UCPs

In Fig. 17, we present the schedulability ratio and the transmission energy consumption under varying number of UCPs to analyze schedulability-ratio-partitioning. We vary the number of UCPs from 16 to 64. We fix the network radius to 4000 m and the number of nodes in a set to 600. In Fig. 17(a), we observe that all control loop sets are schedulable under RTPL-WFUI, RTPL-WFD, RTPL-FFD, RTPL-BFUI, and RTPL-FFUI even with as few as 16 UCPs, due to the use of TSCH-like approach. However, traditional LoRaWAN fails to schedule any control loop sets with 16 UCPs. We observed that LoRaWAN experiences a large number of collisions since adaptive data rate (ADR) randomly changes the transmission parameters such as SF. Even as the number of UCPs is increased, traditional LoRaWAN remains unable to schedule any control loop sets due to high collisions. Additionally,

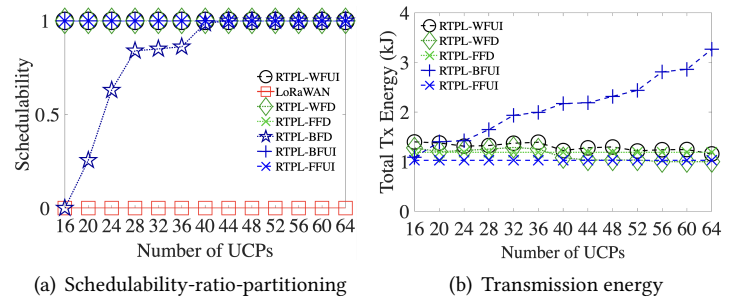


Fig. 17. Results under varying number of UCPs.

we observe that all control loop sets are schedulable under RTPL-WFUI, RTPL-WFD, RTPL-FFD, RTPL-BFUI, and RTPL-FFUI even with as few as 16 UCPs, due to the use of TSCH-like approach. However, traditional LoRaWAN fails to schedule any control loop sets with 16 UCPs. We observed that LoRaWAN experiences a large number of collisions since adaptive data rate (ADR) randomly changes the transmission parameters such as SF. Even as the number of UCPs is increased, traditional LoRaWAN remains unable to schedule any control loop sets due to high collisions. Additionally,

RTPL-BFD fails to schedule any control loop sets with 16 UCPs as it tries to pack one UCP completely before going to the next one. However, its schedulability ratio improves with an increasing number of UCPs, reaching 0.3 with 20 UCPs indicating only 30% of total control loop sets are schedulable. Similarly, it has a schedulability ratio of 0.65 with 24 UCPs, 0.84 with 28 UCPs, and finally 1 with 40 UCPs and more i.e., all control loop sets are schedulable.

In Fig. 17(b), we report the total energy consumption of all the variants of RTPL. We observe that there is only a slight change in energy consumption of other variants except RTPL-BFUI as the number of UCP increases. This is because the best-fit strategy tries to pack one UCP completely before moving to the next one, without taking into consideration the remaining capacity. In RTPL-BFUI, each loop is assigned to the UCP where the WCET of loops is highest, leading to higher energy consumption. On the other hand, we observe that the energy consumption of other variants remains almost constant, with a very slight decrease as the number of UCPs increases. We observed that under smaller number of UCPs, our partition algorithm tries to balance the remaining capacity across all the UCPs, consuming slightly higher energy in comparison to a higher UCP availability. Overall, our partitioning approach allowed us to achieve low energy consumption (under 1.2 kJ) for RTPL-WFUI, RTPL-WEF, RTPL-FFD and RTPL-FFUI variants.

9.5 Results under Varying Duty-cycle

In this simulation, we evaluate our approach under duty-cycle constraints. We fix the number of loops within a set to 200 and the number of UCPs to 28. We set the network radius to 4000 m. Under this setup, we vary the duty-cycle of each UCP from 0.02% to 0.1% and show the Schedulability-ratio-partitioning and transmission energy consumption results in Fig. 18. Fig. 18(a), we observe that RTPL heuristics variants such as RTPL-BFD, RTPL-WFD and RTPL-FFD fail to schedule all control loop sets at low duty cycles. However, RTPL-WFUI is able to achieve close to 100% schedulability ratio. On the contrary, traditional LoRaWAN is not able to schedule any control loops under any duty cycle.

In Fig. 18(b), we observe a constant transmission energy among all variants of RTPL as the duty-cycle increases from 0.02% to 0.1%. However, the transmission energy consumption for RTPL-BFUI increases with an increase in duty-cycle since RTPL-BFUI assigns the loop to the UCP where the WCET of

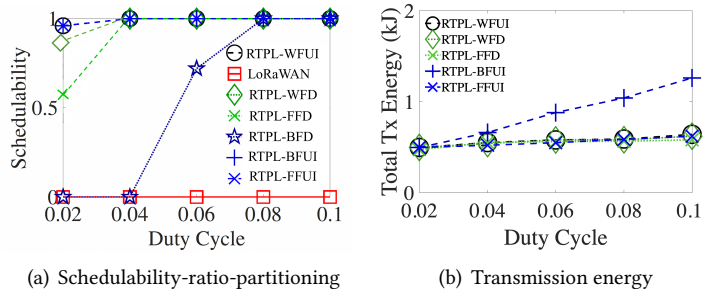


Fig. 18. Results under varying duty-cycle.

loops are highest. Overall, this result shows that under duty-cycle constraints, our partitioning approach is able to provide high schedulability while keeping low energy consumption overhead.

9.6 Comparison of Schedulability under Partitioning and Analysis

In this simulation, we analyze the schedulability of control loops based on the sufficient condition for partitionability (Eq. 8). We report the schedulability ratio as *schedulability-ratio-analysis* for the RTPL-WFUI partitioning heuristic, as it is one of the best partitioning heuristics as observed from the results in previous sections. Leveraging the schedulability analysis allows us to quickly assess the schedulability of control loops by just checking the partitionability condition and obviating the

need to actually run the complete partitioning algorithm each time. Therefore, we compare the results with schedulability-ratio-partitioning.

9.6.1 Results under varying number of UCPs. In this simulation, we fix the number of control loop in a set to 200, the radius to 4000 m and vary the number of UCPs from 16 to 64. We report the schedulability-ratio-analysis and compare the results with the schedulability-ratio-partitioning. In Fig. 19, we observe that none of the control loop sets are schedulable on low number of UCPs (16, 20, 24, 28) under schedulability condition analysis. However, as we increase the number of UCPs, schedulability ratio increases. We observe a sharp increase in schedulability ratio from 0.10 at 36 UCPs to 0.63 at 40 UCPs. We ultimately achieve a schedulability ratio of 1 as we further increase the UCPs to 56 and more.

On the contrary, we observe all the control loop sets are schedulable on all UCPs under the schedulability based on partitioning. This difference in performance between schedulability-ratio-partitioning and schedulability-ratio-analysis is because our proposed schedulability condition is slightly pessimistic. However, it is still significant as reduces the energy consumption significantly by obviating the need of running complete partitioning repeatedly to evaluate schedulability.

9.6.2 Results under varying network radius. In Fig. 20, we report the schedulability ratio for schedulability-ratio-analysis and schedulability-ratio-partitioning under varying network radius. We fix the number of control loops in a set to 200, the number of UCPs to 64 and vary the network radius from 1000 m to 7000 m. We observe all control loops sets are schedulable under schedulability-ratio-analysis at all network radius. This shows that a change in network radius does not impact the performance of our proposed schedulability analysis. On the other hand, schedulability ratio decreases as we increase the network radius under schedulability-ratio-partitioning, declining to 0.9 at 7000 m.

9.6.3 Results under varying number of control loops. In Fig. 21, we fix the number of UCPs to 64, the radius to 4000 m and vary the number of control loops (in a set) from 50 to 400. We observe all the control loops are schedulable under the schedulability-ratio-analysis on 64 UCPs for a set of upto 250 control loops. However, the schedulability ratio decreases as the number of control loops are increased. The schedulability ratio decreases from 0.66 for a set of 300 loops to 0.06 for a set of 350 loops, and eventually reaches 0 for a set of 400 loops. This implies that none of the control loop sets were deemed schedulable under schedulability condition analysis for larger number of control loop sets. On the other hand, we observe all control loop sets are

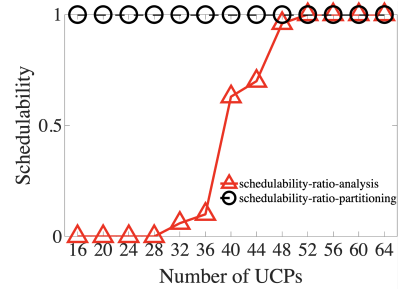


Fig. 19. Schedulability ratio under varying number of UCPs

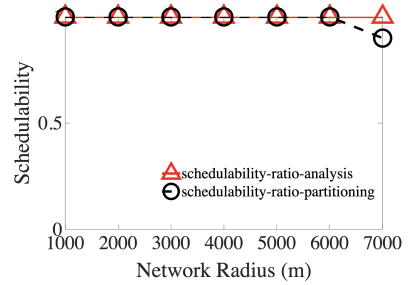


Fig. 20. Schedulability ratio under varying network radius

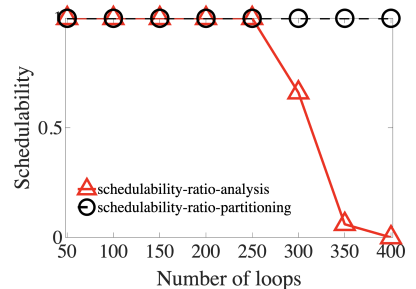


Fig. 21. Schedulability ratio under varying number of loops

schedulable under schedulability based on partitioning. The schedulability condition is slightly pessimistic and therefore the schedulability-ratio-analysis is lower than the schedulability-ratio-partitioning.

9.7 Comparison of Schedulability of RTPL against an Upper Bound

We compare the performance of RTPL-WFUI with respect to an upper bound. Since we defined our partitioning algorithm using worst-fit-UCP-increasing heuristic, we choose RTPL-WFUI to evaluate the performance against an upper bound and refer to it as RTPL in this section.

We use a brute force algorithm to generate an upper bound since an optimal solution is unknown for the partitioning problem for control loops in LoRaWAN. The brute force algorithm explores every possible partition. It returns schedulable if the partition meets the schedulability constraint given in (2) within each UCP/DCP pair and reliability constraint of every control loop (i.e., transmission on the assigned spreading factor results in an acceptable SNIR for decoding). Otherwise, it returns unschedulable. Note that we define the upper bound with respect to the schedulability condition described in the paper since relying on actual simulations to generate a schedulability takes an extremely long time, even for a small network.

Since we use a brute force algorithm, we limit the control loops to 20 with nodes placed randomly in a radius of 1.5 km. We also limit the maximum number of UCP/DCP pairs to 4, limiting the total possible partitions to 160,000 for one control loop set. To observe the impact of changing traffic patterns, we generate control loop sets with varying total minimum-utilization. We define a minimum-utilization of a control loop as the ratio of the smallest WCET of all UCP/DCP pairs and the period. We define total minimum-utilization as the sum of minimum-utilization of all control loops. To generate a control loop set, a total minimum-utilization is first selected for a control loop. Then, each control loop is assigned a random minimum-utilization whose sum equals the total minimum-utilization. From the minimum-utilization, the period of each control loop is computed.

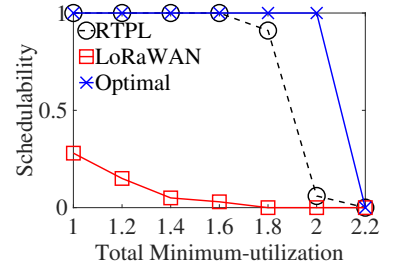


Fig. 22. Results under varying total minimum-utilization.

Fig. 22 reports the schedulability of 100 control loop sets per total minimum-utilization. Fig. 22 shows that RTPL performs similar to the upper bound on partitioning until a total-minimum utilization of 1.8. For a total utilization of 2.0, which is close to the infeasible scenario, the schedulability ratio of RTPL drops to 0.6 while optimal partitioning remains at 1. We observed that ordering the control loops in the increasing number of UCPs was responsible for the drop in the schedulability ratio. However, this drop in performance is observed when the total minimum-utilization approaches the infeasible region (2.2 in this simulation). This simulation shows that RTPL schedules most control loops, except for those with utilization close to the infeasible boundary.

10 CONCLUSION

Traditional wireless solutions for industrial automation depend on short-range wireless technologies (WirelessHART, ISA100.11a), posing a big challenge to support the scale of today's IIoT. To address this limitation, we propose to adopt LoRaWAN, a prominent low-power wide-area network technology, for industrial automation. Adopting LoRaWAN for industrial automation poses some unique challenges. This paper develops RTPL, a real-time scheduling framework for LoRaWAN to enable industrial automation. RTPL overcomes challenges arising from the asymmetric uplink and downlink communication and balancing real-time guarantees and energy constraints. All

results show that RTPL achieves on average 75% improvement in real-time performance without sacrificing throughput or energy compared to traditional LoRaWAN. In future, we aim to extend this work by conducting large-scale LoRaWAN test bed experiments. Additionally, we leave scope to further investigate the potential for deriving a tighter schedulability condition and address the security aspect of RTPL. The issues related to the coexistence of multiple LoRa networks will be studied in the future.

11 ACKNOWLEDGEMENTS

This work was supported by NSF through grants CNS-2301757, CAREER- 2306486, CNS-2306745, and by ONR through grant N00014-23-1-2151.

REFERENCES

- [1] <http://www.dash7-alliance.org>.
- [2] <https://www.u-blox.com/en/lte-cat-m1>.
- [3] https://lora-alliance.org/wp-content/uploads/2020/11/lorawan_regional_parameters_v1.0.3rev0.pdf.
- [4] <http://www.dragino.com/products/lora/item/106-lora-gps-hat.html>.
- [5] <https://lora-developers.semtech.com/library/tech-papers-and-guides/lorawan-class-b-devices/>.
- [6] https://lora-developers.semtech.com/?ACT=72&fid=30&aid=48_0znCpZpvImL3agza59hG&board_id=1.
- [7] https://lora-developers.semtech.com/uploads/documents/files/Understanding_LoRa_Adaptive_Data_Rate_Downloadable.pdf.
- [8] <https://github.com/aakriti-jain/RTPL-extn.git>.
- [9] IEEE 802.11. <http://www.ieee802.org/11>.
- [10] ISA100: Wireless systems for automation. <http://www.isa.org/MSTemplate.cfm?MicrositeID=1134&CommitteeID=6891>.
- [11] LoRa modem design guide. https://www.semtech.com/uploads/documents/LoraDesignGuide_STD.pdf.
- [12] LoRaWAN. <https://www.lora-alliance.org>.
- [13] Lorawan technical report. <https://lora-developers.semtech.com/documentation/tech-papers-and-guides/lora-and-lorawan/>.
- [14] Orthogonality. <https://lora-developers.semtech.com/documentation/tech-papers-and-guides/lora-and-lorawan/>.
- [15] SIGFOX. <http://sigfox.com>.
- [16] The things network. <https://www.thethingsnetwork.org/>.
- [17] WirelessHART system engineering guide. <https://www.emerson.com/documents/automation/emerson-wirelesshart-system-engineering-guide-en-41252.pdf>.
- [18] WirelessHART, 2007. <https://www.fieldcommgroup.org/technologies/hart>.
- [19] Industrial iot trends: Wsn, lpwan & cloud platforms, 2017. https://www.automation.com/pdf_articles/ON_world/InTechMagazineInsert_ONWorldFinal.pdf.
- [20] Karsten Albers and Frank Slomka. An event stream driven approximation for the analysis of real-time systems. In *Proceedings of the 16th Euromicro Conference on Real-Time Systems*, ECRTS '04, pages 187–195, 2004.
- [21] S. Baruah and N. Fisher. The partitioned multiprocessor scheduling of sporadic task systems. In *26th IEEE International Real-Time Systems Symposium (RTSS'05)*, pages 320–329, 2005.
- [22] Sanjoy Baruah, A.K. Mok, and L.E. Rosier. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *RTSS '90*.
- [23] Fatma Benkhelifa, Yathreb Bouazizi, and Julie A. McCann. How orthogonal is lora modulation? *IEEE Internet of Things Journal*, 9(20):19928–19944, 2022.
- [24] Laksh Bhatia, Ivana Tomić, Anqi Fu, Michael Breza, and Julie A McCann. Control communication co-design for wide area cyber-physical systems. *ACM Transactions on Cyber-Physical Systems*, 5(2):1–27, 2021.
- [25] G. C. Buttazzo. *Hard Real-Time Computing Systems*. Springer, 2005. 2nd edition.
- [26] E.P. de Freitas D. Agnoletto, M. Jonsson. Time slot transmission scheme with packet prioritization for bluetooth low energy devices used in real-time applications. *International Journal of Wireless Information Networks*, 27, 2020.
- [27] Adwait Dongare, Revathy Narayanan, Akshay Gadre, Anh Luong, Artur Balanuta, Swarun Kumar, Bob Iannucci, and Anthony Rowe. Charm: Exploiting geographical diversity through coherent combining in low-power wide-area networks. In *Proceedings of the 17th ACM/IEEE International Conference on Information Processing in Sensor Networks*, IPSN '18, pages 60–71, 2018.
- [28] Rashad Eletreby, Diana Zhang, Swarun Kumar, and Osman Yağan. Empowering low-power wide area networks in urban settings. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '17, pages 309–321, 2017.

- [29] Sezana Fahmida, Venkata P Modekurthy, Mahbubur Rahman, Abusayeed Saifullah, and Marco Brocanelli. Long-lived lora: Prolonging the lifetime of a lora network. In *2020 IEEE 28th International Conference on Network Protocols (ICNP)*, pages 1–12. IEEE, 2020.
- [30] East Texas Oil Field. https://en.wikipedia.org/wiki/East_Texas_Oil_Field.
- [31] Mareca Hatler, Darryl Gurganious, and Jeff Kreegar. *Industrial LPWAN – A Market Dynamics Report*. ON World, Inc., November 2018. <https://www.onworld.com/iLPWAN/index.html>.
- [32] Quy Lam Hoang, Woo-Sung Jung, Taehyun Yoon, Daeseung Yoo, and Hoon Oh. A real-time lora protocol for industrial monitoring and control systems. *IEEE Access*, 8:44727–44738, 2020.
- [33] Aakriti Jain, Prashant Modekurthy, and Abusayeed Saifullah. Control over low-power wide-area networks. In *2024 ACM/IEEE 15th International Conference on Cyber-Physical Systems (ICCPs)*, pages 192–201, Los Alamitos, CA, USA, may 2024. IEEE Computer Society.
- [34] David S. Johnson. Fast algorithms for bin packing. *J. Comput. Syst. Sci.*, 8(3):272–314, jun 1974.
- [35] Luca Leonardi, Filippo Battaglia, and Lucia Lo 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*, 6(6):10812–10823, 2019.
- [36] Yan-Ting Liu, Bo-Yi Lin, Xiao-Feng Yue, Zong-Xuan Cai, Zi-Xian Yang, Wei-Hong Liu, Song-Yi Huang, Jun-Lin Lu, Jing-Wen Peng, and Jen-Yeu Chen. A solar powered long range real-time water quality monitoring system by lorawan. In *2018 27th Wireless and Optical Communication Conference (WOCC)*, pages 1–2, 2018.
- [37] C. Lu, A. Saifullah, B. Li, M. Sha, H. Gonzalez, D. Gunatilaka, C. Wu, L. Nie, and Y. Chen. Real-time wireless sensor-actuator networks for industrial cyber-physical systems. *Proceedings of the IEEE*, 104(5):1013–1024, 2016.
- [38] Davide Magrin, Martina Capuzzo, Andrea Zanella, Lorenzo Vangelista, and Michele Zorzi. Performance analysis of lorawan in industrial scenarios. *IEEE Transactions on Industrial Informatics*, 17(9):6241–6250, 2021.
- [39] Aamir Mahmood, Emiliano Ge Sisinni, Lakshmikanth Guntupalli, Raul Rondon, Syed Ali Hassan, and Mikael Gidlund. Scalability analysis of a lora network under imperfect orthogonality. *IEEE Transactions on Industrial Informatics*, 2018.
- [40] Jaco Morné Marais, Reza Malekian, and Adnan M. Abu-Mahfouz. Lora and lorawan testbeds: A review. *2017 IEEE AFRICON*, pages 1496–1501, 2017.
- [41] Paul J Marcelis, Vijay S Rao, and R Venkatesha Prasad. Dare: Data recovery through application layer coding for lorawan. in *2017 IEEE/ACM second international conference on internet-of-things design and implementation (iotdi'17)*, 2017.
- [42] Konstantin Mikhaylov, Juha Petäjälärvi, and Janne Janhunen. On lorawan scalability: Empirical evaluation of susceptibility to inter-network interference. In *2017 European Conference on Networks and Communications (EuCNC)*, pages 1–6, 2017.
- [43] Venkata P Modekurthy, Abusayeed Saifullah, and Sanjay Madria. A distributed real-time scheduling system for industrial wireless networks. *ACM Transactions on Embedded Computing Systems (TECS)*, 20(5):1–28, 2021.
- [44] Venkata Prashant Modekurthy, Dali Ismail, Mahbubur Rahman, and Abusayeed Saifullah. A utilization-based approach for schedulability analysis in wireless control systems. In *2018 IEEE International Conference on Industrial Internet (ICII)*, pages 49–58. IEEE, 2018.
- [45] Hanin Nafi’ah, Puspa Rahmawati, Alfin Hikmaturokhman, and Solichah Larasati. Design of lorawan for smart factories in industrial estates. In *2021 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT)*, pages 116–122, 2021.
- [46] Jorge Ortin, Matteo Cesana, and Alessandro Redondi. Augmenting lorawan performance with listen before talk. *IEEE Transactions on Wireless Communications*, 18(6):3113–3128, 2019.
- [47] J. Parello, B. Claise, and J. Quittek Schoening, B. Energy management framework, 2014. RFC 7326, DOI 10.17487/RFC7326, <https://www.rfc-editor.org/info/rfc7326>.
- [48] Gaetano Patti, Luca Leonardi, and Lucia Lo Bello. A bluetooth low energy real-time protocol for industrial wireless mesh networks. *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, pages 4627–4632, 2016.
- [49] Alessio Pirri, Mattia Pirri, Luca Leonardi, Lucia Lo Bello, and Gaetano Patti. Towards supporting multiple mac protocols on a lorawan end-device for industrial applications. In *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–4, 2023.
- [50] Brecht Reynders, Qing Wang, Pere Tuset-Peiro, Xavier Vilajosana, and Sofie Pollin. Improving reliability and scalability of lorawans through lightweight scheduling. *IEEE Internet of Things Journal*, 2018.
- [51] Abusayeed Saifullah, Dolvara Gunatilaka, Paras Tiwari, Mo Sha, Chenyang Lu, Bo Li, Chengjie Wu, and Yixin Chen. Schedulability analysis under graph routing in WirelessHART networks. In *Proceedings of the IEEE Real-Time Systems Symposium (RTSS)*, pages 165–174, December 2015.
- [52] Abusayeed Saifullah, Mahbubur Rahman, Dali Ismail, Chenyang Lu, Ranveer Chandra, and Jie Liu. Snow: Sensor network over white spaces. In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*, pages 272–285, 2016.

- [53] Govind Sati and Sonika Singh. A review on outdoor propagation models in radio communication. *International Journal of Computer Engineering & Science*, 4(2):64–68, 2014.
- [54] Salahadin Seid, Marco Zennaro, Mulugeta Libsie, Ermanno Pietrosemoli, and Pietro Manzoni. A low cost edge computing and lorawan real time video analytics for road traffic monitoring. In *2020 16th International Conference on Mobility, Sensing and Networking (MSN)*, pages 762–767, 2020.
- [55] Muhammad Osama Shahid, Millan Philipose, Krishna Chintalapudi, Suman Banerjee, and Bhuvana Krishnaswamy. Concurrent interference cancellation: Decoding multi-packet collisions in lora. SIGCOMM '21, page 503–515, New York, NY, USA, 2021.
- [56] Junyang Shi, Mo Sha, and Zhicheng Yang. Digs: Distributed graph routing and scheduling for industrial wireless sensor-actuator networks. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 354–364. IEEE, 2018.
- [57] Shuai Tong, Zhenqiang Xu, and Jiliang Wang. Colora: Enabling multi-packet reception in lora. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, page 2303–2311, 2020.
- [58] Huu Phi Tran, Woo-Sung Jung, Taehyun Yoon, Daeseung Yoo, and Hoon Oh. A two-hop real-time lora protocol for industrial monitoring and control systems. *IEEE Access*, 8:126239–126252, 2020.
- [59] Thiemo Voigt, Martin Bor, Utz Roedig, and Juan Alonso. Mitigating inter-network interference in lora networks. In *Proceedings of the 2017 International Conference on Embedded Wireless Systems and Networks, EWSN '17*, pages 323–328, 2017.
- [60] Xiong Wang, Linghe Kong, Liang He, and Guihai Chen. mlora: A multi-packet reception protocol in lora networks. In *2019 IEEE 27th International Conference on Network Protocols (ICNP)*, pages 1–11, 2019.
- [61] Xianjin Xia, Yuanqing Zheng, and Tao Gu. Ftrack: Parallel decoding for lora transmissions. *IEEE/ACM Transactions on Networking*, 28(6):2573–2586, 2020.
- [62] Zhenqiang Xu, Shuai Tong, Pengjin Xie, and Jiliang Wang. Fliplora: Resolving collisions with up-down quasi-orthogonality. In *2020 17th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 1–9. IEEE, 2020.
- [63] Zhenqiang Xu, Pengjin Xie, and Jiliang Wang. Pyramid: Real-time lora collision decoding with peak tracking. In *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, pages 1–9, 2021.
- [64] Tianyu Zhang, Tao Gong, Chuancai Gu, Huayi Ji, Song Han, Qingxu Deng, and Xiaobo Sharon Hu. Distributed dynamic packet scheduling for handling disturbances in real-time wireless networks. In *2017 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 261–272. IEEE, 2017.
- [65] Dimitrios Zorbas, Khaled Abdelfadeel, Panayiotis Kotzanikolaou, and Dirk Pesch. Ts-lora: Time-slotted lorawan for the industrial internet of things. *Computer Communications*, 153:1–10, 2020.