Real-Time Communication over LoRa Networks

Sezana Fahmida, *Wayne State University*, Detroit, USA
Venkata Prashant Modekurthy, *University of Nevada Las Vegas*, Las Vegas, USA
Dali Ismail, *Southern Illinois University*, Edwardsville, USA
Aakriti Jain & Abusayeed Saifullah, *Iowa State University*, Ames, USA

***Abstract*—Today, industrial Internet of Things (IIoT) are emerging in large-scale and wide-area applications (e.g., oil-field management). 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 LoRa, a prominent low-power wide-area network technology, for industrial automation. Adopting LoRa 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. LoRa usually adopts a simple protocol based on ALOHA with no collision avoidance to minimize energy consumption which is less suitable for real-time communication. Existing real-time protocols for short-range technologies cannot be applied to a LoRa network due to its unique characteristics such as asymmetry between downlink and the uplink spectrum, predefined modes (class) 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 LoRa 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 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 LoRa.**

***Index Terms*—LoRa, Low Power Wide Area Networks, Real-Time, Industrial Internet-of-Things, Closed Loop, Internet-of-Things, Cyber Physical Systems**

I. Introduction

The evolution of Internet of Things (IoT) is transforming the field of industrial automation including process control and smart manufacturing into an important class of Industrial IoT (IIoT). *Industrial automation* entails managing, monitoring, and controlling the production or manufacturing process through sensors and actuators connected over low-bandwidth network for enhanced efficiency and sustainable production. 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 74×8 km^2 requiring tens of thousands of sensors and actuators for automated management [1]. Emerson is targeting to deploy 10,000 nodes (sensors and actuators) for managing an oil-field in Texas [2], [3]. Today, wireless solutions for industrial automation are based on WirelessHART [4] and ISA100.11a [5] 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.

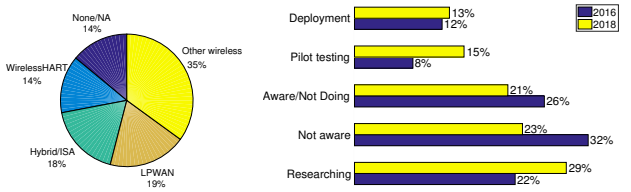
In this paper, we propose to adopt the *Low-Power Wide-Area Network (LPWAN)* technologies for industrial automation. As an emerging IoT technology, LPWAN enables low-power (milliwatts) wireless devices to transmit at low data rates (kbps) over long distances (kms) using narrowband (kHz), thereby obviating the need of multihop and allowing the devices to directly communicate with the control node (gateway). Recently, multiple LPWAN technologies such as LoRa (Long Range) [6], SigFox [7], RPMA [8], DASH7 [9], Telensa [10], NB-IoT [11], LTE Cat M1 [12],and SNOW (Sensor Network Over White spaces) [13]–[18] have appeared. Many of them (e.g., LoRa, SigFox, SNOW) allow numerous devices to concurrently communicate with the gateway, providing high scalability. While the LPWANs have been mainly explored for uplink communication, their potential for industrial automation has not yet been explored.

IIoT realization through LPWAN can greatly benefit the industrial automation solutions like pipeline management [19], 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 multikilometer 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 [20]. 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 [22] and is com-

This work was supported by NSF through grants CNS-2211510, CAREER-2211523, CCF-2118202, and by ONR through grant N00014-22-1-2155.

978-1-6654-9641-4/22/$31.00 ©2022 IEEE
DOI 10.1109/IoTDI54339.2022.00019



(a) Planned Approach (b) LPWAN Adoption Status

Fig. 1. LPWAN adoption trend in industrial automation [20], [21]

mercially available all around the globe with more than 600 known use cases and over 50 million devices deployed [6]. Industry analyst ABI research projects that more than 50% of all LPWAN connections will be based on LoRa by 2026 as it is flexible for both outdoor and indoor cases [23].

Adopting LoRa for industrial automation poses some evolutionarily 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 [24], [25]. For example, tanks in oil-fields need real-time monitoring and control to avoid overflow. 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 based on ALOHA with *no collision avoidance* that is naturally *unsuited* for real-time communication. 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 LoRa. The asymmetry between the downlink and uplink spectrum in a LoRa network also makes it difficult to adopt traditional real-time scheduling techniques.

In this paper, we address the above challenges and propose ***RTPL***, a Real-Time communication Protocol for LoRa networks which can ensure real-time guarantees of end-to-end communication without affecting the concurrent reception capability of LoRa. RTPL closes the control loops over the asymmetric spectrum between uplink and downlink communication and also exploits LoRa's capability of parallel communication. It is a low-overhead real-time MAC protocol allowing autonomous communication of the low-energy devices. 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 LoRa.

The rest of the paper is organized as follows. Section II presents an overview of LoRa. Section III describes the system model. Section IV reviews related work. Section V motivates the need and identifies the challenges for a real-time framework. Section VI presents RTPL. Section VII, VIII and IX presents implementation, experiment results, and simulation results, respectively. Section X concludes the paper.

II. LoRa Overview

LoRa is a pioneering LPWAN technology for connecting sensors to the Cloud. A typical LoRa 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 via a single-hop wireless link.

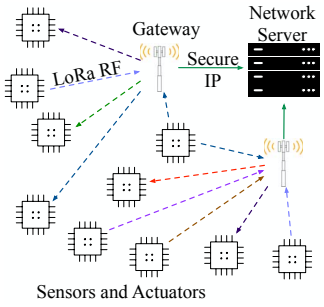


Fig. 2. The LoRa network architecture.

LoRa 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.

LoRa adopts a proprietary Chirp Spread Spectrum (CSS) for communication [23]. 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. There is no duty-cycle requirement on spectrum usage for LoRa in the US. 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 bitrate, and hence, the longer the communication time. LoRa signal data rate is given by $\text{SF} * \frac{4/(4+\text{CR})}{2^{\text{SF}}/\text{BW}}$. Depending on the SF and bandwidth, LoRa achieves data rates between 0.3 kbps and 27 kbps. The SFs in LoRa are orthogonal [26]. Concurrent transmissions with different SF on the same channel do not interfere with each other and can be successfully decoded.

LoRaWAN (LoRa Wide Area Network) is the currently used MAC of a LoRa network. In LoRaWAN, LoRa 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, only closed when transmitting. Class C mode consumes higher power but offers the lowest latency communication from the gateway.

III. System Model

We consider an LPWAN based on LoRa. A LoRa node is equipped with one half-duplex radio, enabling transmission or reception of a packet at a time, but not both. In a LoRa network, a channel and a SF together is called a *communication path*. Two communications paths with different channels or different SFs are *orthogonal* to each other. Concurrent transmissions on orthogonal communication paths do not interfere one another. The number of orthogonal communication paths is a system parameter that depends on vendor specific circuit. The issues related to the coexistence of multiple LoRa networks are out of the scope and will be studied in the future.

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_i, c_2, $\cdots$, c_m and DCPs are denoted as $\delta_1, \delta_2, \cdots \delta_{m'}$. 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.

For industrial automation, LPWAN node can be a sensor or an actuator. Unlike the nodes, the gateway is Internet-connected, line-powered, and is powerful in computation. The nodes can be deployed within several kilometers of the gateway and still communicate directly with it and vice versa. We consider an LPWAN with n wireless control loops $\ell_1, \ell_2, \cdots, \ell_n$, where ℓ_i is the control loop between a sensor node s_i and actuator a_i through a controller at the gateway. 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. 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 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.

IV. Related Work

Real-time scheduling for wireless network for short range technologies such as IEEE 802.15.4 standard was studied widely [27]–[38]. However, these approaches cannot be applied to a LoRa network due to its unique characteristics such as (a) asymmetric communication paths between uplink and downlink, (b) nodes' restriction to operate in one of the three modes (A, B, or C) with severe energy constraints, and (c) concurrent reception capability by exploiting orthogonal SFs.

Existing work on LoRa mostly focuses on improving performance [39]–[41], energy consumption and coverage [42], [43], scalability [22], [44], [45], and packet collision resolution [46]–[51]. The potential of LoRa in real-time communication and control has been explored recently [52]–[55]. Specifically, the work in [55] provides a time-slotted MAC layer on top of LoRaWAN to improve reliability. The work in [52], [56] discusses the feasibility of using LoRa in real-time communications through some measurement studies. The work in [53] proposes a centralized approach to enable both real-time and non real-time traffic in a LoRa network and evaluates through simulations without any implementation on LoRa devices. It considers a simplified model of traffic where all nodes in the network operate on the same transmission interval. As already pointed out in [57], this approach is not applicable to a real industrial IoT network as it does not rely on any slot-scheduling mechanism to meet the real-time requirements of nodes. Besides, this work does not consider the scheduling of downlink actuator commands, and is not applicable to control. 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 [54] to enable event-triggered control. However, this work relaxes the energy constraints as the actuators as need to always remain in continuous listen mode. It also does not take 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/receive data.

V. Feasibility of Closed Loop Real-Time Communication with LoRaWAN

Long-range and single-hop communication of LoRa facilitates low-latency communication. However, LoRa's MAC uses

an ALOHA-based protocol 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 IV, there is no existing work on closed-loop conflict-free MAC protocol for real-time communication that takes into account LoRa's characteristics. Hence, in this section, we experimentally investigate the following: (1) Is real-time communication feasible betweeen sensors and gateway? (2) Is closed-loop real-time communication feasible in LoRa? and (3) Can LoRa adopt existing wireless sensor-actuator network (WSAN) protocols?

A. Can LoRaWAN Ensure Real-Time Guarantees for Uplink Communication?

To answer this question, we ran an experiment with 4 Raspberry Pi 3 with LoRa HAT from Dragino [58] as LoRa sensor nodes and one Dragino LG308 LoRaWAN gateway with The Things Stack network server [59]. The nodes transmitted 10-byte packets with a period of 7 seconds (which is also the packet's deadline) 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 [60]. 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 while the network load was quite low. The results thus show that LoRaWAN can be naturally unsuited for real-time communication.

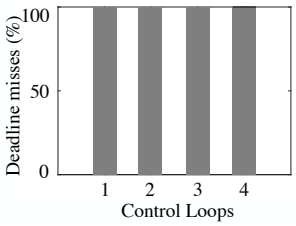


Fig. 3. Percentage of deadline misses.

B. Is Closed-loop Real-Time Communication Feasible under LoRaWAN?

Feedback control loops are the fundamental building blocks in many IIoT and industrial CPS applications. 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.

Since there is no theoretical estimation of uplink latency for LoRa, we perform experiments to observe the latency distribution of packets in uplink communication.We used a similar experimental setup to that in the previous section. 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. In 50% cases, the latency is < 7.4 seconds, while in 90% it is < 7.8 seconds. This variation in delays is caused mainly for two reasons: packet collisions and the variation in the spreading factor used for transmissions. This result shows that estimating the latency of LoRa uplink communication is highly challenging. Therefore, it is not quite feasible to schedule the downlink transmissions using Class B due to unpredictability of the preceding uplink communication of the same control loop under deadline constraints on end-to-end communication. Using Class C for downlink is also not a practical option for energy constrained LoRa nodes as Class C mode has extremely high energy overhead [61]. This result shows that closing feedback loops with real-time communication in a LoRa network needs a new approach.

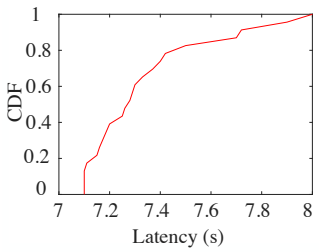


Fig. 4. Latency variation in LoRaWAN.

C. Can we Adopt Existing WSAN MAC Protocols for LoRa?

Real-time communication protocols have been proposed for industrial WSAN based on short-range wireless technologies like WirelessHART [4] and ISA100.11a [5]. 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 LoRa. A LoRa gateway uses asymmetric uplink and downlink spectrum in terms of band, bandwidth, and spreading factors. Furthermore, a LoRa gateway can receive multiple packets simultaneously, and it uses multiple radios to transmit and receive packets concurrently. Thus, existing real-time communication protocols for industrial WSAN are not directly applicable to LoRa. Enabling real-time communication over a LoRa network for industrial automation needs a new approach.

VI. Design of RTPL

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

A. 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)

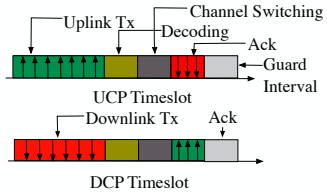


Fig. 5. Uplink and downlink time slot.

like approach which can provide predictable latency and is preferred in industrial automation [4], [5]. While time slotted communication needs time synchronization, in RTPL it can be obtained through time-synchronized beacons in Class B mode. In this mode, the gateway broadcasts time-synchronized beacons periodically (typically 128s to preserve energy). The nodes (sensors and actuators) demodulate the beacon and align their internal clocks with the network server. Based on the network timing reference, the device can receive downlink messages from the network server at scheduled intervals.

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 (maximum packet-size supported by higher spreading factors) packet transmission and its acknowledgment using spreading factor 7 (or 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 $T_{tx}(\text{SF7}, 125 \text{ kHz})$ and transmission time of acknowledgments using spreading factor 7 and 125 kHz bandwidth by $T_{ack}(\text{SF7}, 125 \text{ kHz})$. 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).

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

The different components used in the computation of the time slot are illustrated in Fig. 5. 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. 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 takes 2 time slots and that on spreading factor 10 takes 8 time slots.

If there are m UCPs then m nodes can transmit together to the gateway. The gateway can send control commands to all m nodes through a single broadcast message. When $n > m$, packets need to be prioritized. Unlike CPU task scheduling [62], 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.

A global scheduling approach 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. The partitioning will be done at the gateway. This greatly simplifies the real-time communication protocol and reduces overhead. For example, a node 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. Time synchronization will need to be maintained only inside a partition instead of network-wide global synchronization.

B. Control Loop Partitioning in RTPL

We adopt the concept of partitioned scheduling in LPWAN as follows. The control loops are partitioned into disjoint groups. Each group is assigned a unique UCP and DCP; their uplink and downlink transmissions happen on that UCP and DCP, respectively. The gateway knows the periods and deadlines of all nodes. Assigning the control loops to the UCP and DCP is called *partitioning*. Loops in one partition do not interfere those in another. In TSCH, a transmission starts in the beginning of a slot and ends that slot. Hence, we adopt a *preemptive scheduling* approach, simplifying scheduling.

In each partition, priorities to access the UCP and DCP can be 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. Upon partitioning, the EDF policy on a UCP schedules a packet having the earliest absolute deadline. We break ties based on predetermined policy, such as first come first serve or lower index first, distributed to nodes during network deployment. Since EDF is optimal for preemptive scheduling on a single processor [62], it can offer high schedulability. Thus, we consider partitioned EDF scheduling for RTPL.

C. Challenges of Partitioning in RTPL

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

First, multiprocessor task partitioning assumes that a task's execution requirement remains same on all processors. However, in LoRa, 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 WSAN technologies [4]. 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 [62], 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.

D. 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 LoRa 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 $r_i(c_k, \delta_k) = 2\alpha_i(c_k, \delta_k)$.

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 [63]. 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 $\Omega_i \subseteq \Omega$, where Ω 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 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 EDF uniprocessor schedulability test given by:

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

Note that the EDF uniprocessor schedulability test 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 EDF uniprocessor schedulability 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.

Algorithm 1 Worst-fit-UCP-increasing

Input: Set of all control loops L, Set of all UCP, DCP pair,Ω
Output: Set of Partitions $\{\pi(c_k, \delta_k) \quad k = 1, 2, \cdots, m\}$
for $l_i \in L$ **do**
  $\Omega_i \leftarrow$ ComputeAvailableCP(l_i)
end for
$L' \leftarrow$ sort L in increasing order of $|\Omega_i|$
for $l_i \in L'$ **do**
  assigned $\leftarrow$ FALSE
  for (c_k, δ_k) in Ω_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
  $P' \leftarrow$ sort Ω_i in decreasing order of ρ
  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

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. Furthermore, our approach can be extended to SigFox which is similar to LoRa.

E. Handling Workload and Network Dynamics

To meet application requirement, the period of a control loop may change over the network operation time. In case of such workload dynamics, the utilization of a control loop may increase. Note that the partitioning heuristic for RTPL will typically balance the remaining capacity across all UCP, DCP pairs. As such, small variations of workload may be handled without re-executing the partitioning heuristic if the current partition is still schedulable under the new utilization of the loop. Thus, the network server can notify the other loops in the partition regarding the workload change and the loops can update the schedule accordingly. However, in the worst-case, if the workload changes significantly, the partitioning heuristic is executed again. Similarly, if some channel becomes overly noisy, i.e, signal-to-noise (SNR) ratio becomes very low, the

gateway can blacklist it and re-run the partitioning algorithm excluding that channel.

F. Handling Different Numbers of UCPs and DCPs

Typically, regional regulations limit the number of DCPs to be less than the number of UCPs. For example, in the US the number of downlink channels is 8, while the number of uplink channels is 64. Since DCPs are enabled through separate radios, a high number of DCPs become expensive. One approach to address this limitation can be to use only m' UCPs in the uplink partition, which can lead to under-utilizatiion of the network.

In RTPL, we group control messages of several control loops together into a single downlink communication message. The grouping of multiple packets modifies 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 $\pi(c_i, c_j, \cdots c_k, \delta_p)$ where $\Phi(c_i) = \Phi(c_j) = \cdots = \Phi(c_k) = \Phi(\delta_p)$. 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 gateways 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.

Although packet grouping is known to increase the number of time slots required by the gateway, we have observed that such overhead is minimal for LoRa. The low time alot overheads is due to the non-uniform bandwidths allocated for uplink and downlink communications. Typically, Lora uses a 125 kHz bandwidth for uplink communication and a 500 kHz bandwidth for downlink communication. The higher bandwidth for downlink communication reduces the airtime by 74%. 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 [64]. Thus, LoRa enables the grouping of 4 downlink packets with minimal overhead. Note that depending on the grouping selected, the number of time slots allocated to the sensor may be different from that allocated to the actuator of the same control loop. 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 $r_i(c_i, c_j, \cdots c_k, \delta_p) = 2\alpha_i(c_i, c_j, \cdots c_k, \delta_p) = 2\max\left(r_i^{up}(c_i, c_j, \cdots c_k, \delta_p), r_i^{down}(c_i, c_j, \cdots c_k, \delta_p)\right)$. Using the new grouping and WCET calculation, the worst-fit-UCP-increasing heuristic can be used to generate the partitions.

G. Handling Channel Hopping and Duty-Cycling

In the US, there is a maximum dwell time regulation of 400ms on any channel. Additionally, if a node utilizes at least 50 uplink channels, the nodes are required to implement pseudo-random channel hopping before every transmission. Note that the LoRa nodes are exempt from the channel hopping regulation if the output power is limited to 21dBm and only 8 uplink channels are utilized [65]. To comply with the maximum dwell time regulation, we refrain from using SF>10. We can easily extend our framework to handle the channel hopping regulation by having a predetermined sequence of frequency hopping for each UCP. The gateway shares the sequence with the nodes in the partition. After each time slot, the node hops to another channel as per the predetermined sequence. Through hopping, the physical channel of a UCP changes but the partitioning remains unchanged as all nodes within a partition switch to the same channel.

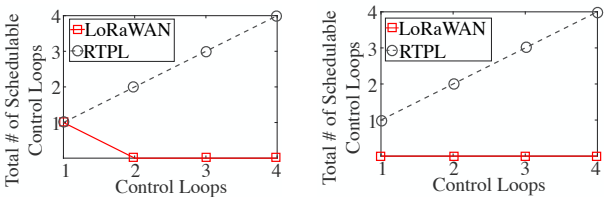
Many regions (e.g., North America) do not have any duty-cycle requirement. In such regions, a LoRa network can host relatively more workload and applications with real-time requirements. However, there are duty-cycle regulations in some regions (e.g, in Europe, duty cycle is set to 1%) to ensure fair usage of the spectrum. RTPL can be easily extended to handle duty-cycling for such regions. Specifically, the partitioning approach in RTPL has to be changed to handle the duty-cycle regulation by stalling the UCP/DCP pair with dummy control loops. Namely, for every UCP/DCP pair with duty-cycle d, we assign a dummy control loop with utilization $(1 - d)$. 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.

VII. Implementation

We have implemented RTPL using Raspberry Pi LoRa HAT (LoRaHAT) and Dragino LG308 LoRaWAN gateway (GW) as LoRa nodes and gateway, respectively [58]. 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 [23]. 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 [66]. We use the Things Stack network server from The Things Industries [59]. It is an open source and widely used LoRaWAN network server. Due to the lack of a working implementation of LoRa 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. In our experiments, we store the EDF schedule locally at each node. We assigned a time slot of 7 seconds based on (1). Note that this includes a 5 second receive window delay recommended by the The Things Stack for sending an acknowledgment.

VIII. 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 IV, there are some



(a) Harmonic periods (b) Non-harmonic periods

Fig. 6. Schedulability of RTPL

existing works that explore the potential of LoRa in enabling real-time control [53], [54]. However, the work in [53] does not provide a slot-scheduling algorithm to meet real-time requirements. While, [54] considers an event-triggered model through a contention-based communication which is not directly applicable to time-triggered systems requiring deadline guaranties. Hence, we evaluate the performance of RTPL by comparing it with LoRaWAN through physical experiments with up to four control loops. We plan to setup a larger testbed and perform large scale physical experiments in the future.

A. Default Parameters

We use the following default parameters setting for all experiments unless stated otherwise.

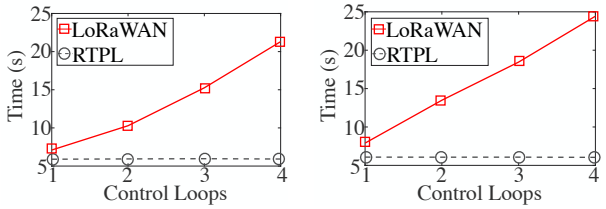
- Period: 7, 14, 56, 112 (seconds)
- SF: 7, 8, 9, 10
- Deadline = period
- Frequencies: 903.9-904.7 MHz
- Packet size: 10 bytes
- Bandwidth: Uplink: 125 kHz, Downlink: 500 kHz
- TX power: 15 dBm
- Distance: Indoor: 2 m - 6 m, Outdoor: 200 m - 500 m

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 ADR algorithm.

B. Indoor Deployment

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 default parameters for the experiments in this Section.

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 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



(a) Avg latency per loop (b) Maximum latency per loop

Fig. 7. Latency vs. # of control loops

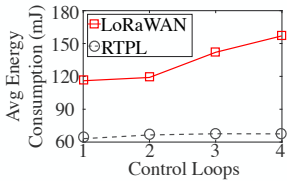


Fig. 8. Avg energy consumption

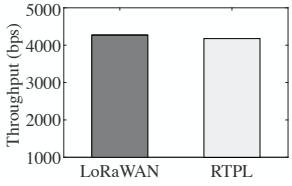


Fig. 9. Throughput (indoor)

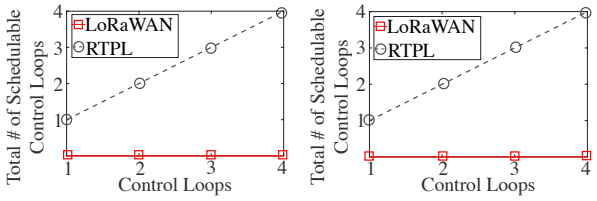
RTPL, while one control loop is schedulable under LoRaWAN. This is due to LoRaWAN adaptive data rate (ADR) mechanism that controls several parameters (e.g., SF). Fig. 6(b) shows that 100% of the control loops are schedulable with RTPL using a non-harmonic period. This is due to RTPL partitioning, which guarantees each packet meets its deadline. For LoRaWAN, 100% of the control loops are unschedulable.

3) Energy and Latency: In this Section, we measure the end-to-end energy consumption and 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. Note that we 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 seconds). 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 seconds. This is due to collisions and the acknowledgment mechanism in LoRaWAN, (the GW acknowledges uplink transmissions sequentially).

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.

4) Throughput: To compare the throughput of RTPL and LoRaWAN, we need measure the throughput of the control loop without considering its schedulability. For example, if a packet missed its deadline and is successfully received by

Fig. 10. Outdoor GW and nodes locations



(a) Harmonic periods (b) Non-harmonic periods

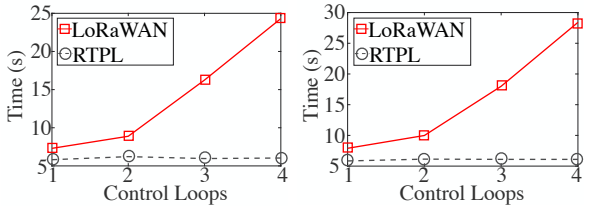
Fig. 11. Schedulability of RTPL (outdoor)

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 compared 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.

C. Outdoor Deployment

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, all the experiments use the default setting unless mentioned otherwise.

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.



(a) Avg latency per loop (b) Maximum latency per loop

Fig. 12. Latency vs. # of control loops

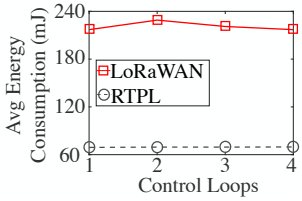


Fig. 13. Avg energy consumption

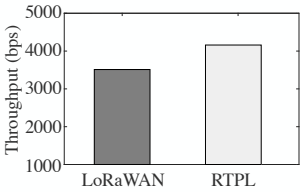


Fig. 14. Throughput (outdoor)

3) Energy and Latency: We estimate the end-to-end energy consumption and 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. 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 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 also measure the average energy consumption per control loop for the outdoor scenario. The result in Fig. 13 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 result shows that distance has an insignificant impact on the energy consumption of the control loop. Also, RTPL provides schedulability guarantees while remaining energy-efficient.

4) Throughput: 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, respectively. This result shows that RTPL achieves better throughput in outdoor deployments than LoRaWAN.

IX. Simulation

Here, we evaluate RTPL with large-scale simulations in NS3 with a single gateway and up to 1000 control loops.

A. Setup

We use the LoRaWAN NS3 module proposed in [67] for our simulation with up to 52 UCPs and 18 DCPs. Each control loop was assigned a random period chosen from the range $[2^{12}, 2^{25}]$ ms and the maximum number of retransmission was set as 2 time slots. We used a timeslot length of 115 ms. Sensors and actuators of the loops were positioned randomly across a disc centered at the gateway with radius up to 6

km. We define *schedulability ratio* as the ratio of number of schedulable control loop sets to total control loop sets. We report the schedulability ratio and total transmission energy consumption for the schedulable cases. Each result is averaged over 25 simulation runs of 1 hour. We compare our approach with regular LoRaWAN and the following variations of RTPL:

BFD: variation of RTPL following the best-fit decreasing heuristic, where the nodes are ordered in decreasing order of utilization and the UCP with the least remaining capacity is chosen first for assigning a loop.

WFD: variation of RTPL following the worst-fit decreasing heuristic. Loops are ordered in decreasing order of utilization.

FFD: 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.

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

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

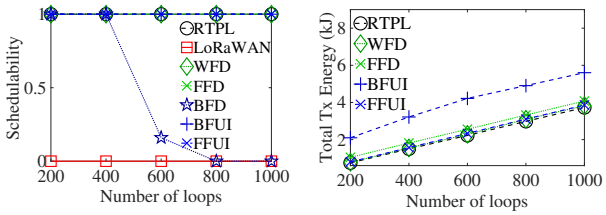
B. Results under Varying Number of Control Loops

In this simulation, we fix the number of UCPs to 40 and the network radius to 4 km and vary the number of control loops from 200 to 1000. Fig. 15 shows the schedulability ratio and transmission energy consumption under varying number of nodes for RTPL and the baseline approaches. In Fig. 15(a), 100% of the control loop sets are schedulable under RTPL, while in traditional LoRaWAN, none of the control loop sets are schedulable. This is due to high number of collisions in regular LoRa. Furthermore, we see that the schedulability of the RTPL variant using traditional BFD becomes close to 0 at only 600 nodes. We noticed that a best-fit strategy assigns a loop to the UCP where the WCET of that loop becomes highest, as it results in the lowest remaining capacity on that UCP. This, along with the fact that traditional BFD does not consider the number of available UCP/DCP pair for a loop while assigning it to a partition, results in poor schedulability performance of best fit. On the contrary, RTPL, which uses Worst-Fit UCP-Increasing, is highly scalable.

In Fig. 15(b), we show the total transmission energy consumption for the schedulable control loop sets under the same setup. Note that since traditional LoRaWAN and BFD are not schedulable for all cases, we do not show the energy consumption for these approaches. In this figure, the energy consumption for all approaches steadily increase as the number of control loops increase. However, RTPL has the lowest energy consumption in the compared approaches. In fact, it has on average 45% less energy consumption than BFUI, showing the energy-efficiency of our partitioning heuristic.

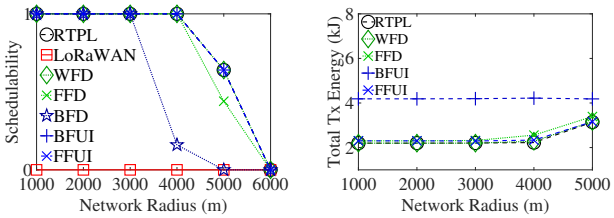
C. Results under Varying Network Radius

In this simulation, we set the number of control loops to 600, the number of UCPs to 40 and vary the network radius from 1000 m to 6000 m and show the result in Fig. 16. In Fig. 16(a), we see a sharp decline in schedulability for



(a) Schedulability (b) Transmission energy

Fig. 15. Results under varying number of nodes.



(a) Schedulability (b) Transmission Energy

Fig. 16. Results under varying network radius.

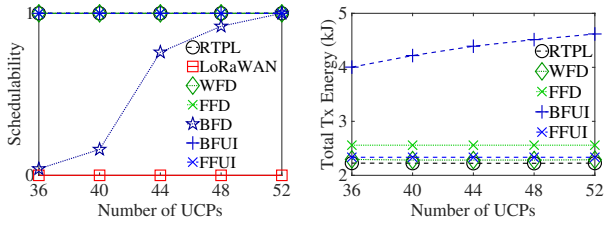
all partitioning approaches as the network radius increases to 6000 m. At such a large network radius, the control loops are required use the higher SFs to ensure a successful packet reception at the gateway, leading to limited number of available UCP/DCP pairs. This leads to unschedulable cases at higher network radius. However, we observe that RTPL has the highest schedulability ratio among all approaches up to the network radius of 5000 m.

In Fig. 16(b), we notice that the total transmission energy consumption for almost all schedulable partitioning approaches remain similar until the network radius of 4000m. However at radius 5000 m, we see a sharp increase in energy consumption for all approaches. This is due to the nodes using higher SFs at increased network radius. Furthermore, as BFUI tries to assign loops to UCPs with the highest available SF by default, it's energy consumption remains similar at different network radius. However, we note that RTPL consistently has the lowest energy consumption.

D. Results under Varying Number of UCPs

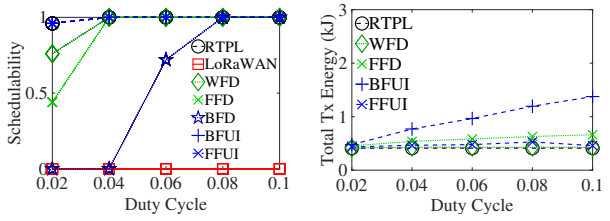
In Fig. 17, we fix the network radius to 4000 m, the number of nodes to 600 and vary the number of UCPs from 36 to 52. In Fig. 17(a), we notice that at lower number of UCPs, traditional BFD heuristic fails to schedule all control loops. However, the balanced partitioning stategy used in RTPL is able to achieve 100% schedulability ratio even with just 36 UCPs. On the other hand, LoRaWAN is unschedulable even with 52 UCPs due to high number of collisions.

We report the total transmission energy for the same setup in Fig. 17(b). In this figure, we see that as the number of UCPs increase, the total energy consumption of RTPL does not change much, as the partitioning heuristic attempts to equally balance the remaining capacity accross all UCPs. However, in the case of the best-fit variation of RTPL (BFUI), the total



(a) Schedulability (b) Transmission Energy

Fig. 17. Results under varying number of UCPs.



(a) Schedulability (b) Transmission Energy

Fig. 18. Results under varying duty-cycle.

energy consumption increases with the number of UCPs. With higher number of UCPs, loops are assigned to UCPs where the WCET of loops are highest, leading to a high energy consumption for BFUI. This shows the limitation of best-fit based approaches in terms of energy consumption.

E. Results under Varying Duty-cycle

In this simulation, we evaluate our approach under duty-cycle constraints. We fix the number of loops to 200, the network radius was set to 4000 m and the number of UCPs was set to 36. The control loops used a random sampling period chosen from the range $[2^{15}, 2^{25}]$ ms. Under this setup we vary the duty-cycle of each UCP from 0.02 to 0.1 and show the results in Fig. 18.

In Fig. 18(a), we notice that traditional bin-packing heuristics such as BFD,WFD and FFD are not able to schedule all control loops at low duty cycles. However, RTPL is able to achieve close to 100% schedulability ratio in all cases. In Fig. 18(b), we see that RTPL has the lowest transmission energy consumption among all approaches. We also notice that as the duty-cycle increases from 0.02 to 0.1, transmission energy consumption for BFUI increases. This is due to each loop being assigned to the UCP where the WCET of loops are highest. Overall, this result shows that RTPL is able to provide high schedulability and low energy consumption overhead under duty-cycle constraints.

F. Comparison of RTPL against an Upper Bound

We compare the performance of RTPL with respect to an upper bound. 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 LoRa. 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. 19 reports the schedulability of 100 control loop sets per total minimum-utilization. Fig. 19 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.

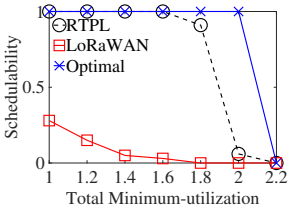


Fig. 19. Results under varying total minimum-utilization.

X. 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 LoRa, a prominent low-power wide-area network technology, for industrial automation. Adopting LoRa for industrial automation poses some unique challenges. This paper develops RTPL, a real-time scheduling framework for LoRa 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 LoRa.

REFERENCES

[1] E. T. O. Field, https://en.wikipedia.org/wiki/East_Texas_Oil_Field.

[2] "WirelessHART system engineering guide," https://www.emerson.com/documents/automation/emerson-wirelesshart-system-engineering-guide-en-41252.pdf.

[3] http://www.automationworld.com/networking-amp-connectivity/wireless-shines-emerson-global-user-exchange.

[4] "WirelessHART," 2007, https://www.fieldcommgroup.org/technologies/hart.

[5] "ISA100: Wireless systems for automation," http://www.isa.org/MSTemplate.cfm?MicrositeID=1134&CommitteeID=6891.

[6] "LoRaWAN," https://www.lora-alliance.org.

[7] "SIGFOX," http://sigfox.com.

[8] https://www.ingenu.com/technology/rpma.

[9] http://www.dash7-alliance.org.

[10] "Telensa," https://www.telensa.com.

[11] https://www.u-blox.com/en/narrowband-iot-nb-iot.

[12] https://www.u-blox.com/en/lte-cat-m1.

[13] A. Saifullah, M. Rahman, D. Ismail, C. Lu, J. Liu, and R. Chandra, "Low-power wide-area networks over white spaces," *ACM/IEEE Transactions on Networking*, vol. 26, no. 4, pp. 1893–1906, 2018.

[14] A. Saifullah, M. Rahman, D. Ismail, C. Lu, R. Chandra, and J. Liu, "Snow: Sensor network over white spaces," in *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*, 2016, pp. 272–285.

[15] A. Saifullah, M. Rahman, D. Ismail, C. Lu, J. Liu, and R. Chandra, "Enabling reliable, asynchronous, and bidirectional communication in sensor networks over white spaces," in *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, 2017, pp. 1–14.

[16] M. Rahman, D. Ismail, V. P. Modekurthy, and A. Saifullah, "Implementation of lpwan over white spaces for practical deployment," in *Proceedings of the International Conference on Internet of Things Design and Implementation*, 2019, pp. 178–189.

[17] ——, "Lpwan in the tv white spaces: A practical implementation and deployment experiences," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 20, no. 4, pp. 1–26, 2021.

[18] V. P. Modekurthy, D. Ismail, M. Rahman, and A. Saifullah, "Low-latency in-band integration of multiple low-power wide-area networks," in *2021 IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2021, pp. 333–346.

[19] J. Parello, B. Claise, and a. J. Q. Schoening, B., "Energy management framework," 2014, rFC 7326, DOI 10.17487/RFC7326, https://www.rfc-editor.org/info/rfc7326.

[20] M. Hatler, D. Gurganious, and J. Kreegar, *Industrial LPWAN – A Market Dynamics Report*. ON World, Inc., November 2018, https://www.onworld.com/iLPWAN/index.html.

[21] "Industrial iot trends: Wsn, lpwan & cloud platforms," 2017, https://www.automation.com/pdf_articles/ON_world/InTechMagazineInsert_ONWorldFinal.pdf.

[22] P. J. Marcelis, V. S. Rao, and R. V. 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.

[23] "LoRa modem design guide," https://www.semtech.com/uploads/documents/LoraDesignGuide_STD.pdf.

[24] J. Stankovic, T. Abdelzaher, C. Lu, L. Sha, and J. Hou, "Real-time communication and coordination in embedded sensor networks," *Proceedings of the IEEE*, vol. 91, no. 7, pp. 1002–1022, 2003.

[25] 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*, vol. 104, no. 5, pp. 1013–1024, 2016.

[26] "Lorawan technical report," https://lora-developers.semtech.com/documentation/tech-papers-and-guides/lora-and-lorawan/.

[27] X. Jin, N. Guan, C. Xia, J. Wang, and P. Zeng, "Packet aggregation real-time scheduling for large-scale wia-pa industrial wireless sensor networks," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 17, no. 5, p. 88, 2018.

[28] T. Zhang, T. Gong, C. Gu, H. Ji, S. Han, Q. Deng, and X. S. 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)*. IEEE, 2017, pp. 261–272.

[29] T. Zhang, T. Gong, Z. Yun, S. Han, Q. Deng, and X. S. Hu, "Fd-pas: A fully distributed packet scheduling framework for handling disturbances in real-time wireless networks," in *2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2018, pp. 1–12.

[30] T. Zhang, T. Gong, S. Han, Q. Deng, and X. S. Hu, "Distributed dynamic packet scheduling framework for handling disturbances in real-time wireless networks," *IEEE Transactions on Mobile Computing*, 2018.

[31] J. Shi, M. Sha, and Z. Yang, "Digs: Distributed graph routing and scheduling for industrial wireless sensor-actuator networks," in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2018, pp. 354–364.

[32] ——, "Distributed graph routing and scheduling for industrial wireless sensor-actuator networks," *IEEE/ACM Transactions on Networking*, vol. 27, no. 4, 2019.

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

[34] ——, "Distributed graph routing for wirelesshart networks," in *Proceedings of the 19th International Conference on Distributed Computing and Networking*, 2018, pp. 1–10.

[35] A. Saifullah, D. Gunatilaka, P. Tiwari, M. Sha, C. Lu, B. Li, C. Wu, and Y. Chen, "Schedulability analysis under graph routing in WirelessHART networks," in *Proceedings of the IEEE Real-Time Systems Symposium (RTSS)*, December 2015, pp. 165–174.

[36] V. P. Modekurthy, A. Saifullah, and S. Madria, "A distributed real-time scheduling system for industrial wireless networks," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 20, no. 5, pp. 1–28, 2021.

[37] V. P. Modekurthy, D. Ismail, M. Rahman, and A. Saifullah, "A utilization-based approach for schedulability analysis in wireless control systems," in *2018 IEEE International Conference on Industrial Internet (ICII)*. IEEE, 2018, pp. 49–58.

[38] V. P. Modekurthy and A. Saifullah, "Online period selection for wireless control systems," in *2019 IEEE International Conference on Industrial Internet (ICII)*. IEEE, 2019, pp. 170–179.

[39] R. Eletreby, D. Zhang, S. Kumar, and O. Yağan, "Empowering low-power wide area networks in urban settings," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '17, 2017, pp. 309–321.

[40] R. Oliveira, L. Guardalben, and S. Sargento, "Long range communications in urban and rural environments," in *2017 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2017, pp. 810–817.

[41] T. Voigt, M. Bor, U. Roedig, and J. Alonso, "Mitigating inter-network interference in lora networks," in *Proceedings of the 2017 International Conference on Embedded Wireless Systems and Networks*, ser. EWSN '17, 2017, pp. 323–328.

[42] A. Dongare, R. Narayanan, A. Gadre, A. Luong, A. Balanuta, S. Kumar, B. Iannucci, and A. 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*, ser. IPSN '18, 2018, pp. 60–71.

[43] S. Fahmida, V. P. Modekurthy, M. Rahman, A. Saifullah, and M. Brocanelli, "Long-lived lora: Prolonging the lifetime of a lora network," in *2020 IEEE 28th International Conference on Network Protocols (ICNP)*. IEEE, 2020, pp. 1–12.

[44] A. Mahmood, E. G. Sisinni, L. Guntupalli, R. Rondon, S. A. Hassan, and M. Gidlund, "Scalability analysis of a lora network under imperfect orthogonality," *IEEE Transactions on Industrial Informatics*, 2018.

[45] B. Reynders, Q. Wang, P. Tuset-Peiro, X. Vilajosana, and S. Pollin, "Improving reliability and scalability of lorawans through lightweight scheduling," *IEEE Internet of Things Journal*, 2018.

[46] Z. Xu, P. Xie, and J. Wang, "Pyramid: Real-time lora collision decoding with peak tracking," in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, 2021, pp. 1–9.

[47] X. Xia, Y. Zheng, and T. Gu, "Ftrack: Parallel decoding for lora transmissions," *IEEE/ACM Transactions on Networking*, vol. 28, no. 6, pp. 2573–2586, 2020.

[48] M. O. Shahid, M. Philipose, K. Chintalapudi, S. Banerjee, and B. Krishnaswamy, "Concurrent interference cancellation: Decoding multi-packet collisions in lora," ser. SIGCOMM '21, New York, NY, USA, 2021, p. 503–515.

[49] X. Wang, L. Kong, L. He, and G. Chen, "mlora: A multi-packet reception protocol in lora networks," in *2019 IEEE 27th International Conference on Network Protocols (ICNP)*, 2019, pp. 1–11.
[50] S. Tong, Z. Xu, and J. Wang, "Colora: Enabling multi-packet reception in lora," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 2020, p. 2303–2311.
[51] Z. Xu, S. Tong, P. Xie, and J. Wang, "Fliplora: Resolving collisions with up-down quasi-orthogonality," in *2020 17th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, 2020, pp. 1–9.
[52] M. Rizzi, P. Ferrari, A. Flammini, and E. Sisinni, "Evaluation of the iot lorawan solution for distributed measurement applications," *IEEE Transactions on Instrumentation and Measurement*, vol. 66, no. 12, pp. 3340–3349, 2017.
[53] L. Leonardi, F. Battaglia, and L. L. Bello, "Rt-lora: A medium access strategy to support real-time flows over lora-based networks for industrial iot applications," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10 812–10 823, 2019.
[54] L. Bhatia, I. Tomić, A. Fu, M. Breza, and J. A. McCann, "Control communication co-design for wide area cyber-physical systems," *ACM Transactions on Cyber-Physical Systems*, vol. 5, no. 2, pp. 1–27, 2021.
[55] D. Zorbas, K. Abdelfadeel, P. Kotzanikolaou, and D. Pesch, "Ts-lora: Time-slotted lorawan for the industrial internet of things," *Computer Communications*, vol. 153, pp. 1–10, 2020.
[56] S. Liu, C. Xia, and Z. Zhao, "A low-power real-time air quality monitoring system using lpwan based on lora," in *2016 13th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*, 2016, pp. 379–381.
[57] Q. L. Hoang, W.-S. Jung, T. Yoon, D. Yoo, and H. Oh, "A real-time lora protocol for industrial monitoring and control systems," *IEEE Access*, vol. 8, pp. 44 727–44 738, 2020.
[58] http://www.dragino.com/products/lora/item/106-lora-gps-hat.html.
[59] "The things stack," https://www.thethingsindustries.com/stack/.
[60] "Major changes in the things stack," https://www.thethingsindustries.com/docs/getting-started/migrating/major-changes/.
[61] https://lora-developers.semtech.com/library/tech-papers-and-guides/lorawan-class-b-devices/.
[62] G. C. Buttazzo, *Hard Real-Time Computing Systems*. Springer, 2005, 2nd edition.
[63] G. Sati and S. Singh, "A review on outdoor propagation models in radio communication," *International Journal of Computer Engineering & Science*, vol. 4, no. 2, pp. 64–68, 2014.
[64] https://www.thethingsnetwork.org/airtime-calculator.
[65] "Lorawan regional parameters," https://lora-alliance.org/wp-content/uploads/2020/11/rp_2-1.0.1.pdf/.
[66] "Lorawan regional parameters," https://www.dragino.com/products/lora-lorawan-gateway/item/140-lg308.html.
[67] D. Magrin, M. Centenaro, and L. Vangelista, "Performance evaluation of lora networks in a smart city scenario," in *2017 IEEE International Conference on communications (ICC)*. ieee, 2017, pp. 1–7.