

# Latency-Optimized Data Harvesting in LoRa Networks Using a Mobile Sink

Abusayeed Saifullah, Nasif Ahmed, Aakriti Jain  
*Department of Computer Science*  
*University of Texas at Dallas*  
Richardson, TX, USA

Md Yusuf Sarwar Uddin  
*Department of Computer Science*  
*University of Missouri-Kansas City*  
Kansas City, MO, USA

Mohammad Ashiqur Rahman  
*School of Comp. and Inf. Sciences*  
*Florida International University*  
Miami, FL, USA

**Abstract**—Remote IoT deployments in challenging terrains increasingly rely on LoRa and mobile sinks (e.g., drones) for data harvesting. Nodes produce heterogeneous, time-varying, high-volume data that must be uploaded within a brief contact window as the mobile sink traverses the network. Therefore, a critical requirement for data harvesting in remote LoRa deployments is to optimize latency in collecting data when a mobile sink arrives. Existing work mostly focuses on LoRa networks with in situ gateways; optimizing data harvesting latency with mobile sinks remains an unaddressed challenge. In this paper, we propose a novel data harvesting protocol that achieves near-optimal latency in remote LoRa deployments using a mobile sink equipped with a LoRa gateway. Our protocol leverages the parallel reception capabilities of a LoRa gateway by grouping nodes to confine collisions within each group, each group being assigned a unique channel-spreading factor pair. To optimize network latency under varying data volumes across nodes, we formulate node grouping as an optimization problem, prove its NP-hardness, and propose a novel 2-approximation algorithm, which the protocol adopts on the fly for concurrent transmission scheduling across groups. This work presents the first provable constant approximation bound on latency for a LoRa network. Our small-scale experiments show that the latency achieved through our approach can be as close as 1.07 times the optimal latency. Additionally, large-scale simulations in NS-3 demonstrate that our approach reduces latency by up to 80% compared to a state-of-the-art approach, while also decreasing energy consumption by 30%.

## I. INTRODUCTION

The proliferation of the Internet of Things (IoT) has led to an explosion in connected devices, generating vast amounts of data across diverse and often remote environments. This paradigm shift necessitates communication technologies that offer extensive coverage, minimal energy consumption, and low infrastructure costs. Low-Power Wide Area Networks (LPWANs) have emerged as a key enabler for such large-scale IoT deployments, allowing energy-constrained devices to transmit data over distances of several kilometers. Among these, LoRa (Long Range) [1] stands out as a leading LPWAN technology, boasting over 600 documented use cases such as health monitoring [2], forest fire detection [3], sailing tracking [4], and precision agriculture [5], [6], and more than 50 million deployed devices globally [7], [8].

Many IoT applications involve large-scale deployments in remote or geographically challenging environments, such as forests, deserts, mountains, or offshore platforms, where maintaining fixed gateways or reliable cloud connectivity can be

difficult. In such scenarios, mobile sinks (also known as data mules), such as drones, trains, or satellites, are employed to collect data from sensor nodes as they traverse the deployment area, a process known as *data harvesting*. This approach is common in applications where historical data is needed for offline analysis and study of long-term trends, such as forest fire monitoring, agricultural/environmental monitoring, and structural health monitoring. In these settings, sensor nodes may collect data at varying intervals periodically, sporadically, or in response to specific events. Namely, different nodes can sense different phenomena or respond to varying event-driven triggers. For example, in agricultural applications, soil moisture may vary gradually while pest activity triggers short-term, high-frequency sensing. As a result, nodes can possess heterogeneous amounts of data at any given time, and the same node may generate varying amounts of data over time.

While these data may not be time-sensitive, their collection via a mobile sink becomes time-critical because nodes have limited time to transmit while the sink traverses. For example, a train cannot stop along its route, and a drone is constrained by limited battery life and must complete its mission before exhaustion. Therefore, a critical requirement for data harvesting in remote LoRa deployments is to optimize latency in collecting data when a mobile sink arrives. LoRaWAN, the default Medium Access Control (MAC) protocol for LoRa, employs ALOHA, a simple random access protocol without collision avoidance, making it inherently unsuitable to collect a huge amount of data from numerous nodes within a short timeframe. Although Carrier Sense Multiple Access (CSMA) and Listen-Before-Talk (LBT) can reduce collisions under moderate loads, they require significant energy for continuous channel sensing and introduce substantial delays. Pure Time Division Multiple Access (TDMA) involves synchronization and scheduling overheads that become impractical at large scales or in dynamic traffic conditions. Most existing research focuses on LoRa networks with in situ gateways; no prior work has addressed optimizing latency for data harvesting in remote LoRa deployments utilizing mobile sinks.

In this paper, we address remote LoRa deployments and propose a novel data harvesting protocol with near-optimal latency using a mobile sink equipped with a LoRa gateway. Leveraging the parallel reception capabilities of a LoRa gateway on multiple channels and spreading factors (a LoRa pa-

parameter that controls the number of bits encoded per symbol), our protocol employs a grouping of nodes, each group being assigned a unique pair of channel and spreading factor. This approach confines collisions within each group, thereby enabling parallel transmission scheduling across different groups. To optimize network latency under varying amounts of data across nodes, we formulate the grouping as an optimization problem that aims to balance the workload among groups. We prove that the problem is *NP-hard*. Then we propose a novel *2-approximation algorithm* to solve this grouping problem, which the protocol adopts for transmission scheduling to minimize the network latency. To handle traffic dynamics, it performs grouping of  $n$  nodes *on-the-fly* with a computational complexity of  $O(n^2)$ . We theoretically prove its approximation ratio of 2 on network latency. To our knowledge, this is the first ever work with a provable constant approximation bound on latency for a LoRa network.

Finally, we evaluate the protocol through both real-world implementation and simulations using NS-3. Our small-scale experiments show that the latency achieved through our approach can be as close as 1.07 times the optimal latency. Large-scale simulations demonstrate that our approach reduces latency by up to 80% compared to a state-of-the-art approach [9], while also decreasing energy consumption by 30%.

The remainder of this paper is organized as follows. Section II reviews the related work. Section III describes the background and our system model. Section IV details the proposed data harvesting protocol. Sections V and VI present the experimental and simulation results, respectively. Finally, Section VII concludes the paper.

## II. RELATED WORK

Existing research on LoRa primarily focuses on enhancing performance [10]–[13], optimizing energy consumption and coverage [14], [15], improving scalability [16]–[18], or boosting signal-to-noise ratio (SNR) [19]–[21]. Many works have proposed resolving packet collisions [22]–[29]. However, these are physical-layer solutions, and are effective only for limited collisions and become inefficient and energy-intensive in dense networks with heterogeneous traffic.

Contention-based schemes such as CSMA [30], [31], LBT [32], [33], and random backoff mechanisms [34], [35] have been proposed to mitigate collisions. Some works introduce gateway beacons to improve coordination [36]. While these methods can reduce collisions under moderate loads, they require significant energy for continuous channel sensing and introduce substantial delays. TDMA protocols [37]–[39] offer collision-free communication but involve synchronization and scheduling overheads that become impractical for data harvesting at large scales or dynamic traffics. Burst-MAC [9] extends TDMA principles to handle bursty traffic in a LoRa network but relies on a always-on gateway and cannot guarantee near-optimal latency. The work in [40] employs a mobile sink for data harvesting for traditional short-range sensor network aiming to enhance network lifetime. It does not target LoRa-based systems and does not focus on latency optimization.

Thus, existing research focuses on LoRa networks with in situ gateways. No prior work has addressed optimizing latency for data harvesting in remote LoRa deployments utilizing mobile sinks. In contrast, we have proposed a data harvesting protocol for a LoRa network with near-optimal latency using a mobile sink. To the best of our knowledge, this is the first work to establish a provable constant approximation bound on latency for data collection in a LoRa network.

## III. BACKGROUND AND SYSTEM MODEL

First we provide an overview of LoRa and then describe our system model based on a LoRa network for data harvesting.

### A. An Overview of LoRa

LoRa is a pioneering LPWAN technology developed by Semtech, designed for long-range, low-power, and low-data-rate communication in IoT applications. In the LoRa protocol stack, *LoRa* and *LoRaWAN* refer to the physical and link layers, respectively. A typical LoRa network consists of multiple end devices (called *LoRa nodes*), one or more gateways, and a central network server. The nodes, functioning as sensor nodes, communicate wirelessly with the gateways over long distances. The gateways relay the collected data to the network server via a backhaul connection such as Ethernet or cellular links. LoRa uses multiple *uplink* and *downlink* channels, where nodes transmit data to gateways over uplink channels, and gateways respond through downlink channels.

The LoRa physical layer employs *Chirp Spread Spectrum* (CSS) modulation, which spreads the transmitted signal over a wide bandwidth, providing resilience against interference and noise. Each LoRa transceiver has five configurable runtime parameters: *spreading factor*, bandwidth, carrier frequency, transmission power, and coding rate, which jointly determine communication range, reliability, and energy consumption.

In the US, LoRa specifies 64 uplink channels of 125 kHz bandwidth, spaced 200 kHz apart within the 902.3–914.9 MHz band, and 8 downlink channels of 500 kHz bandwidth, spaced 600 kHz apart in the 923.3–927.5 MHz band. The spreading factor, ranging from 7 to 12, controls the number of bits per symbol. Higher spreading factors improve receiver sensitivity and extend range by increasing the SNR, but reduce data rate and increase airtime. The packet airtime doubles with each increment in spreading factor. LoRa employs Orthogonal Variable Spreading Factor (OVSF) technique, which allows concurrent transmissions on the same channel each using a different spreading factor without mutual interference [1]. Unlike Europe with duty-cycle restrictions, the US imposes no explicit duty-cycle limit, but each 125 kHz channel is restricted to a maximum dwell time of 400 ms per transmission.

LoRaWAN employs an ALOHA-based MAC protocol, where nodes send packets whenever they have data without carrier sensing. Nodes can operate in one of three modes: *Class A*, *B*, or *C*. Class A nodes operate asynchronously with bidirectional communication, opening two short downlink windows after each uplink, offering minimal power consumption. Class B nodes add scheduled receive windows

synchronized via periodic beacons, allowing more frequent downlink opportunities for time-sensitive data. Class C nodes keep their receive windows open continuously (except during uplinks), minimizing downlink latency at the cost of higher energy consumption. LoRaWAN also supports *confirmed* and *unconfirmed* messages: confirmed messages request an Acknowledgment (ACK) and allow up to 8 retransmissions for reliability, whereas unconfirmed messages do not require an ACK, saving energy and reducing network traffic.

### B. System Model

We consider the problem of optimizing latency for data harvesting on a remote LoRa deployment, called a *LoRa network* throughout the paper, using a mobile sink equipped with a LoRa gateway. We consider applications that involve large-scale deployments in remote or geographically challenging environments such as forests, deserts, mountains, or offshore platforms where fixed gateways or cloud connectivity are difficult to maintain. In such scenarios, mobile sinks such as drones, trains, or cars, are employed to collect data from sensor nodes as they traverse the deployment area, which is the process known as *data harvesting*. It is common in applications where historical data is needed for offline analysis and study of long-term trends such as forest fire monitoring, environmental monitoring, smart agriculture, structural health monitoring. The nodes may collect data at varying intervals periodically, sporadically, or in response to specific events. As a result, nodes can possess different amounts of data at different time. While these data may not be time-sensitive, their collection via a mobile sink becomes time-critical because nodes have limited time to transmit while the sink passes by. For example, a drone must return to its recharge point before its battery is depleted. Therefore, a critical requirement for data harvesting in remote LoRa deployments is to optimize latency in collecting data when a mobile sink arrives.

The mobile sink periodically visits the network to collect data, and this period is known to the nodes. We consider that it remains in the communication ranges of all nodes during its contact time window. To detect the sink's arrival without continuous monitoring or strict time synchronization, nodes can wake up shortly before the estimated arrival time based on their internal clocks. Alternatively, as the estimated arrival time approaches, nodes can increase their wake-up frequency to sense the mobile sink's presence. The nodes can do duty-cycling as needed for energy-saving. But we do not consider any duty cycling requirement. Nodes may have varying packet counts across different rounds of the sink's arrival and these numbers can vary among nodes. We use  $p_i$  to denote the number of packets node  $v_i, 1 \leq i \leq n$ , has to deliver to the sink in a particular round, where  $n$  is total nodes in the deployment. All packets are of an equal length.

The LoRa gateway can receive on multiple channels concurrently. It can also transmit concurrently on multiple channels. In addition, as stated in the LoRa specification [1], multiple concurrent transmissions can happen on a channel successfully, each using a different spreading factor. While the spread-

ing factors may not be 100% orthogonal in practice, existing studies have confirmed that they are nearly 100% orthogonal [9]. Considering  $m$  uplink channels and  $k$  spreading factors,  $mk$  nodes can transmit to the gateway concurrently, each using a unique pair of a channel and a spreading factor, the pair being called a *virtual channel*. Similarly, with  $m^{\text{down}}$  downlink channels, the gateway can send  $m^{\text{down}}k$  packets concurrently, each on a separate downlink virtual channel. The packet airtime doubles with each increment in spreading factor.

### IV. PROPOSED DATA HARVESTING PROTOCOL

In this section, we describe our proposed data harvesting protocol and theoretically prove its near-optimal performance. Achieving predictable, collision-free communication is challenging, particularly when nodes exhibit dynamic and heterogeneous traffic patterns. Nodes may generate event-driven data, leading to varying packet counts across different rounds of the sink's arrival and heterogeneous amounts of data/packets among nodes. To address this, our protocol operates in two phases. In **Phase I**, the system first gathers traffic information from all nodes using a simple collision-free transmission scheduling. Once the workload of each node is known, the mobile sink orchestrates **Phase II** to collect all packets within the minimum possible time. This is achieved through a greedy policy that guarantees a 2-approximation ratio for latency minimization. Both phases enable conflict-free scheduling through virtual grouping of nodes based on virtual channels, confining potential collisions within each group. The virtual grouping in Phase II is created on the fly to handle the dynamic traffic pattern of the nodes, while that of Phase I can be reused over time, as its only purpose is to collect the traffic information from the nodes. Phase I primarily serves as an initial overhead to enable the main data collection process in Phase II. Specifically, we cannot use Phase I groups for Phase II communications because the groups in two phases are different. Hence, new Phase II groups need to be created. Schedule of Phase II kicks off after the nodes receive this grouping information from the sink.

#### A. Collecting Packet Counts through Phase I Virtual Grouping

1) *Mobile Sink-Node Coordination*: The mobile sink periodically visits the network to collect data and the nodes wake up shortly before its estimated arrival time based on their internal clocks. Alternatively, as the estimated arrival time approaches, nodes can increase their wake-up frequency to sense the mobile sink's presence. Figure 1 illustrates the node's behavior during sink detection. Nodes remain in an idle or sleep state when no data are available for transmission or the sink's arrival is not imminent. When the data is ready and the arrival time nears, nodes monitor their designated channel. If no activity is detected, they return to sleep and recheck after the next scheduled wake-up interval. Upon detecting activity, indicating the sink's presence, nodes prepare for data transmission. The sink initiates its presence by transmitting a downlink beacon. This beacon is sent on a common, known downlink channel with an extended preamble, ensuring its

duration exceeds the nodes' wake-up interval for reliable detection. All nodes monitor this channel to detect the gateway. Upon detection, nodes transition to LoRa Class B operation and continue monitoring the downlink beacon. This beacon also synchronizes the nodes' clocks with the mobile sink, a synchronization that is maintained throughout the data collection process. Upon synchronization, time is slotted. A *time slot* is determined as the time needed for one packet transmission using the smallest spreading factor.

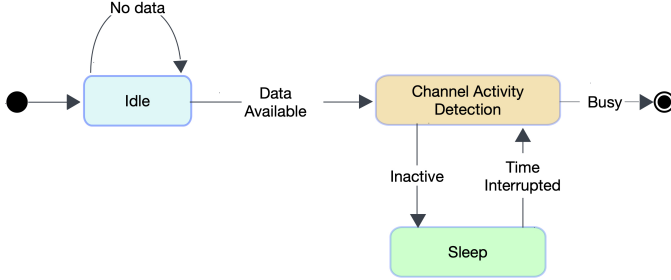


Fig. 1: Node operation modes during mobile sink detection

2) *Collision-Free Uplink Communication*: Once the nodes establish a connection with the mobile sink, each node will need to transmit its total packet count to the sink. To ensure collision-free communication, we employ a virtual grouping strategy designed to contain potential collisions within each group. A *virtual channel* is defined as a unique combination of a physical channel and a spreading factor. Nodes assigned to the same virtual channel form a virtual group. With  $m$  available uplink channels and  $k$  spreading factors, we establish a total of  $mk$  uplink virtual groups<sup>1</sup>, indexed from 1 to  $mk$ . Each virtual group (except possibly the last) in Phase I will contain  $\lceil \frac{n}{mk} \rceil$  nodes. To transmit packet counts, one node from each virtual group transmits in parallel to the mobile sink. Within each group, nodes transmit sequentially. Nodes within each virtual group are assigned a unique relative ID from 1, 2,  $\dots$ ,  $\lceil \frac{n}{mk} \rceil$ , which indicates the order in which they transmit their packet count value. When a node is given its group information a priori, this relative ID is also given to it. If a transmission fails, the affected node is assigned a retransmission slot, which is the next available slot on its designated virtual channel. Retransmissions can be scheduled as many times as needed, up to a predefined maximum. Communicating retransmission slot assignments to nodes presents a challenge, which our approach addresses as described next.

3) *Handling Downlink Transmissions and Network Dynamics*: Retransmission management is communicated through downlink messages. A significant challenge in LoRa is that downlink channels are different from uplink channels, and their number ( $m^{\text{down}}$ ) is smaller than uplink channels (i.e.,  $m^{\text{down}} < m$ ). Therefore, sending downlink message will require us to create another set of virtual groups named *downlink*

<sup>1</sup>To avoid confusion between uplink and downlink virtual groups, by default *virtual group* will mean *uplink virtual group* in both Phase I and Phase II. For downlink, we explicitly use the term *downlink virtual group* in both phases.

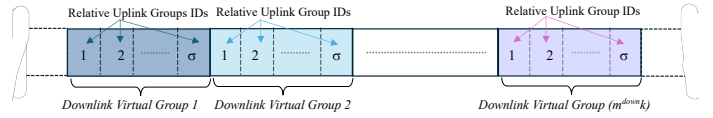


Fig. 2: Downlink frame structure showing fields for receiving nodes.

*virtual groups*. With  $m^{\text{down}}$  downlink channels and  $k$  spreading factors, we create  $m^{\text{down}}k$  downlink virtual groups, indexed from 1 to  $m^{\text{down}}k$ . Each downlink virtual group will have  $\sigma$  uplink virtual groups (except the last one, which may have fewer), each uplink virtual group inside a downlink virtual group having a unique relative uplink group ID 1, 2,  $\dots$ ,  $\sigma$ , where

$$\sigma = \left\lceil \frac{mk}{m^{\text{down}}k} \right\rceil = \left\lceil \frac{m}{m^{\text{down}}} \right\rceil$$

After each slot, the mobile sink will send a downlink transmission with information for each of the nodes whose transmissions completed in the last slot (i.e., whose ACKs are pending). This information can be ACK, retransmission slot assignment, or other grouping or scheduling information based on the type of downlink message. The downlink message will contain  $(m^{\text{down}}k)\sigma$  fields indexed as 1, 2,  $\dots$ ,  $(m^{\text{down}}k)\sigma$ , each field containing information for one node. For a node inside the downlink virtual group having index  $d$  and inside the uplink virtual group having relative index  $u$  (relative virtual group id in that downlink group), the information will be written in the field number:

$$(d-1)\sigma + u. \quad (1)$$

A node accesses its relevant field using its downlink virtual group index and its relative uplink group index. A value of 1 in a field is reserved to indicate an ACK for the node's packet. If a transmission fails, the retransmission slot number is written in this field. Figure 2 illustrates the structure of these fields within a downlink message frame.

Phase I virtual groups (both uplink and downlink) are pre-established before the protocol starts and remain unchanged unless some nodes join or leave the network. If a new node joins, the sink is manually informed and the node is manually assigned to an available virtual uplink group and a virtual downlink group. The new node receives the next available relative ID within its assigned virtual group. If nodes leave the network, their allocated slots may either remain unused, be assigned to new nodes if they join simultaneously, or necessitate a complete re-assignment of virtual uplink and downlink group indexes, with new information provided to nodes via downlink messages. Upon successful completion of Phase I, the mobile sink transmits one or more beacons instructing the nodes to start Phase II communication.

### B. Transmission Scheduling through Phase II Virtual Grouping to Minimize Latency

The virtual grouping performed in Phase I may not be effective for transmission scheduling to collect the packets from the LoRa nodes as the numbers of packets among them

can vary significantly. Namely, some groups may be overly loaded while some may remain underutilized. Therefore, we perform a Phase II grouping of the nodes to assign them to virtual channels based on their packet counts so that the overall latency to deliver all packets of the network is minimized. A node must be assigned to one and only one virtual channel. Nodes assigned to the same virtual channel form a Phase II *virtual group*. In a virtual group, the nodes transmit their packets sequentially in the order they were assigned to that group. That is, the node assigned first will transmit all of its packets first; then the node assigned after that will transmit its packets, and so on. However, this ordering inside a group will not impact its schedule length. The schedule of the packets inside a virtual group is called its *superframe*. The length of a virtual group's superframe depends on the number of nodes, their packet counts, and the assigned spreading factor, and is measured in time slots. Since the transmissions on all virtual channels can happen in parallel, the length of the longest superframe across all virtual groups indicates the *network latency* for delivering all packets to the mobile sink. Note that this latency does not include the time spent in Phase I.

Note that the Phase II virtual grouping we are presenting now is the uplink virtual grouping. Our objective is to create Phase II virtual groups in order to minimize network latency. To take into account the traffic dynamics (i.e., the nodes generating a varying number of packets across rounds), phase II grouping needs to be performed on the fly in every round when the mobile sink arrives and gets connected with the nodes. In what follows, we formulate the Phase II virtual grouping problem, prove its NP-hardness, present a Near-Optimal Virtual grouping Algorithm (NOVA), and prove that NOVA has an approximation ratio of 2. First we present our approach without considering retransmissions. Retransmission handling will be described after that.

1) *Problem Formulation*: The network has a total of  $n$  LoRa nodes denoted by  $V = \{v_i | 1 \leq i \leq n\}$  and a total of  $m$  uplink physical channels (frequencies) denoted by  $\{f_i | 1 \leq i \leq m\}$ . Let the  $k$  spreading factors be denoted by  $\{s_i | 1 \leq i \leq k\}$ . When the mobile sink arrives at the network, let  $p_i$  be the number of packets that node  $v_i$  needs to transmit. There will be a total of  $mk$  virtual channels or virtual groups. A virtual channel is denoted by an ordered pair  $(c, j)$ , where  $f_c, 1 \leq c \leq m$ , is the physical uplink channel and  $s_j, 1 \leq j \leq k$ , is the spreading factor. The set of nodes that are assigned virtual channel  $(c, j)$  form a virtual group denoted by  $V_{c,j}$ . Therefore, assigning a node to a virtual group  $V_{c,j}$  has the same meaning as assigning a node to a virtual channel  $(c, j)$ , and we interchangeably use these two phrases.

We consider all packets are of the same length. One time slot is the length of time needed for transmitting one packet using the lowest spreading factor. In LoRa, the time needed to transmit the same packet using the next larger spreading factor doubles, and so on. Therefore, the transmission time for a packet using spreading factor  $s_j$  is  $2^{j-1}$  time slots.

Our objective is to assign each node  $v_i, 1 \leq i \leq n$ , to a virtual channel  $(c, j), 1 \leq c \leq m, 1 \leq j \leq k$ , to minimize

the network latency denoted by  $L$  to deliver all packets to the mobile sink.

To formulate this problem, we can consider this assignment in terms of a binary decision variable  $x_{i,c,j}$  defined as follows:

$$x_{i,c,j} = \begin{cases} 1, & \text{if node } v_i \text{ is assigned to virtual group } V_{c,j}; \\ 0, & \text{otherwise.} \end{cases}$$

where  $i \in \{1, 2, \dots, n\}$ ,  $c \in \{1, 2, \dots, m\}$ , and  $j \in \{1, 2, \dots, k\}$ . Since each node must be assigned to exactly one virtual channel, the following constraint must be satisfied.

$$\sum_{c=1}^m \sum_{j=1}^k x_{i,c,j} = 1, \quad \forall i \in \{1, 2, \dots, n\}$$

The superframe length for virtual group  $V_{c,j}$  is denoted by  $L_{c,j}$ , and is given by

$$L_{c,j} = \sum_{i=1}^n x_{i,c,j} \cdot p_i \cdot 2^{j-1}.$$

Since the longest superframe length across all virtual channels is the network latency  $L$ ,

$$L = \max \left\{ \sum_{i=1}^n x_{i,c,j} \cdot p_i \cdot 2^{j-1} \mid 1 \leq c \leq m; 1 \leq j \leq k \right\} \quad (2)$$

**P1**: Now the optimal virtual grouping boils down to determining the binary decision variables  $x_{i,c,j} \in \{0, 1\}$ ,  $\forall i \in \{1, 2, \dots, n\}, \forall c \in \{1, 2, \dots, m\}, \forall j \in \{1, 2, \dots, k\}$  to

**Minimize**  $L$

*Subject to*

$$\sum_{c=1}^m \sum_{j=1}^k x_{i,c,j} = 1, \quad 1 \leq i \leq n \quad (3)$$

$$L \geq \sum_{i=1}^n x_{i,c,j} \cdot p_i \cdot 2^{j-1}, \quad 1 \leq c \leq m, 1 \leq j \leq k \quad (4)$$

Constraint (3) in the above formulation ensures that each node must be assigned to exactly one virtual channel. Constraint (4) states that the network latency (i.e., the schedule length) is no less than the total time needed by the nodes assigned to the same virtual channel.

2) *Proof of NP-Hardness*: In this section, we prove that an optimal virtual grouping to minimize latency, as formulated in the above section (Problem **P1**), is NP-hard. Theorem 1 proves the NP-hardness through reduction from the well-known *Partition Problem*. The Partition Problem (Problem **P2**) is defined as follows:

**P2**: Given a set  $A = \{a_1, a_2, \dots, a_{n'}\}$  of  $n'$  positive integers, determine whether there exists a subset  $A'$  of  $A$  such that:

$$\sum_{a_i \in A'} a_i = \sum_{a_i \in A \setminus A'} a_i = \frac{1}{2} \sum_{a_i \in A} a_i$$

The Partition Problem is known to be NP-complete.

**Theorem 1.** *An optimal virtual grouping to minimize network latency in a LoRa network (Problem P1) is NP-hard.*

*Proof:* Given an instance of the Partition Problem,  $A = \{a_1, a_2, \dots, a_n\}$ , we construct an instance of the optimal virtual grouping of a LoRa network as follows.

- Number of nodes:  $n$ ; and each value  $a_i$  corresponds to LoRa node  $v_i$ .
- Number of channels:  $m = 2$ .
- Number of spreading factors:  $k = 1$ .
- $p_i = a_i$  for each node  $v_i$ .
- Since  $k = 1$ , the transmission time for each packet is  $2^{1-1} = 1$  time slot.

The above reduction from the Partition Problem (P2) to the optimal virtual grouping problem (P1) takes  $O(n)$  time.

Now we prove that there is a solution to the Partition Problem (P2) if and only if there is a virtual grouping for the LoRa network with a network latency  $L \leq \frac{1}{2} \sum_{i=1}^n a_i$ .

*If Partition Problem has a solution  $\implies$  Virtual grouping problem has a solution with  $L \leq \frac{1}{2} \sum_{i=1}^n a_i$ :* Suppose there exists a subset  $A'$  of  $A$  such that  $\sum_{a_i \in A'} a_i = \frac{1}{2} \sum_{a_i \in A} a_i$ . We construct a virtual grouping of LoRa nodes as follows:

- Assign nodes corresponding to elements in  $A'$  to virtual group  $V_{1,1}$ , i.e., channel  $f_1$  and spreading factor  $s_1$ .
- Assign nodes corresponding to elements in  $A \setminus A'$  to virtual group  $V_{2,1}$ , i.e., channel  $f_2$ , spreading factor  $s_1$ .

Thus, the superframe length in virtual group  $V_{1,1}$  is

$$\sum_{a_i \in A'} a_i \cdot 1 = \frac{1}{2} \sum_{i=1}^n a_i,$$

and in virtual group  $V_{2,1}$  is

$$\sum_{a_i \in A \setminus A'} a_i \cdot 1 = \frac{1}{2} \sum_{i=1}^n a_i$$

Therefore, the network latency  $L$  is given by  $L = \max\left(\frac{1}{2} \sum_{i=1}^n a_i, \frac{1}{2} \sum_{i=1}^n a_i\right) = \frac{1}{2} \sum_{i=1}^n a_i$ .

Thus,  $L \leq \frac{1}{2} \sum_{i=1}^n a_i$ .

*If Virtual Grouping problem has a solution with  $L \leq \frac{1}{2} \sum_{i=1}^n a_i \implies$  Partition Problem has a solution:* Suppose there exists a virtual grouping yielding network latency  $L \leq \frac{1}{2} \sum_{i=1}^n a_i$ . Since we have only 2 channels and one spreading factor, each node must be assigned to one of the two virtual groups:  $V_{1,1}$ ,  $V_{2,1}$ . Let  $A'$  be the subset of  $A$  corresponding to the nodes assigned to virtual group  $V_{1,1}$ , and let  $A \setminus A'$  be the subset corresponding to the nodes assigned to group  $V_{2,1}$ . Then, constraint (4) of P1 gives us:

$$L \geq \sum_{a_i \in A'} a_i \cdot 1 \quad \text{and} \quad L \geq \sum_{a_i \in A \setminus A'} a_i \cdot 1$$

Since  $L \leq \frac{1}{2} \sum_{i=1}^n a_i$ , we have:

$$\sum_{a_i \in A'} a_i \leq \frac{1}{2} \sum_{i=1}^n a_i \quad \text{and} \quad \sum_{a_i \in A \setminus A'} a_i \leq \frac{1}{2} \sum_{i=1}^n a_i$$

Also, we know that:

$$\sum_{a_i \in A'} a_i + \sum_{a_i \in A \setminus A'} a_i = \sum_{i=1}^n a_i$$

Therefore, the only way to satisfy these inequalities is if:

$$\sum_{a_i \in A'} a_i = \frac{1}{2} \sum_{i=1}^n a_i \quad \text{and} \quad \sum_{a_i \in A \setminus A'} a_i = \frac{1}{2} \sum_{i=1}^n a_i$$

This implies that  $A'$  is a solution to the Partition Problem. Since the Virtual Grouping of nodes in a LoRa network to minimize latency is NP-hard. ■

3) *Near-Optimal Virtual grouping Algorithm (NOVA):* Given the NP-hardness of optimal virtual subgrouping, it is unlikely that a polynomial-time solution exists unless  $P = NP$ . Therefore, we propose a polynomial-time Near-Optimal Virtual grouping Algorithm (NOVA).

NOVA employs an intuitive greedy approach. In each iteration, it assigns a node to a virtual channel, aiming to minimize the maximum superframe length (i.e., latency) across all virtual subgroups established thus far. This greedy strategy is designed to be highly effective in minimizing the overall latency upon completion of all assignments. To make the approach even more affective, we also want to first assign the nodes with higher time demands, i.e., with larger number of packets. Therefore, NOVA first sorts the nodes in non-increasing order of  $p_i$ , the first node having the highest number of packets. For each node  $v_i$ , it evaluates all available virtual channels  $(c, j)$ , and selects the virtual channel that minimizes the completion time of node  $v_i$ . The *completion time* of  $v_i$  is calculated as  $L_{c,j} + p_i \cdot 2^{j-1}$ , where  $L_{c,j}$  is the current superframe length for the nodes assigned so far to virtual channel  $(c, j)$ . Node  $v_i$  is assigned to virtual channel  $(c^*, j^*)$  that yields the minimum completion time. Virtual group  $V_{c^*, j^*}$  is updated by adding  $v_i$  to it, and  $L_{c^*, j^*}$  is then updated by incrementing the time needed by node  $v_i$  as follows:

$$V_{c^*, j^*} = V_{c^*, j^*} \cup \{v_i\};$$

$$L_{c^*, j^*} = L_{c^*, j^*} + p_i \cdot 2^{j^*-1}.$$

Finally, after all nodes have been assigned, the latency,  $L$ , is determined as the longest superframe length across all virtual groups. This represents the total time required to deliver all packets to the mobile sink in the LoRa network. Algorithm 1 outlines the steps of NOVA.

**Algorithm 1:** Near-Optimal Virtual grouping Algorithm (NOVA)

- 1: **Input:**  $n$  LoRa nodes,  $m$  uplink channels,  $k$  spreading factors,  $p_i$  (number of packets for node  $v_i$ );
- 2: **Output:** Uplink virtual groups  $V_{c,j}$ , Latency  $L$ ;
- 3: **Initialize:**  $V_{c,j} = \emptyset$ ,  $\forall c, 1 \leq c \leq m$ ,  $\forall j, 1 \leq j \leq k$ ;
- 4: Create a list of nodes  $V' = \{v_1, v_2, \dots, v_n\}$ ;
- 5: Sort the list  $V'$  in non-increasing order of  $p_i$ ;
- 6: **while**  $V'$  is not empty **do**
- 7:   Remove the first node  $v_i$  from  $V'$ .
- 8:   Find the virtual channel  $(c^*, j^*)$  that minimizes the completion time of node  $v_i$ :

$$(c^*, j^*) = \arg \min_{(c,j)} \{L_{c,j} + p_i \cdot 2^{j-1}\},$$

where  $L_{c,j}$  is the current superframe length of virtual group  $V_{c,j}$ .

- 9:   Assign node  $v_i$  to virtual channel  $(c^*, j^*)$ .
- 10:   Update  $V_{c^*, j^*} = V_{c^*, j^*} \cup \{v_i\}$ ;
- 11:   Update  $L_{c^*, j^*} = L_{c^*, j^*} + p_i \cdot 2^{j^*-1}$ ;
- 12: **end while**
- 13:  $L = \max_{(c,j)} L_{c,j}$ ;
- 14: **return**  $L$  and virtual groups  $V_{c,j}, \forall c, 1 \leq c \leq m, \forall j, 1 \leq j \leq k$ ;

**Complexity Analysis.** The complexity of NOVA can be analyzed as follows. Sorting the list of  $n$  nodes takes  $O(n \log n)$  time. In each iteration of the *while* loop, the algorithm assigns one node to a virtual group. Finding the best virtual channel requires checking all virtual channels, taking  $O(mk)$  time in each iteration. Thus, assignment of all nodes takes  $O(nmk)$  time. Thus, total time:  $O(n \log n + nmk)$ . Since in large-scale deployment, typically  $mk < n$ , the complexity can be approximated as  $O(n \log n + n^2) = O(n^2)$ .

**Downlink Communication for Schedule Distribution and Retransmission Scheduling.** Once Phase II uplink virtual groups are created, we create new downlink virtual groups for Phase II using the same technique as in Phase I. This new grouping becomes effective only after this grouping information is successfully transmitted to the nodes. We pass this information to the nodes through downlink communications using the Phase I uplink and downlink virtual grouping. To send the schedule information, for each node, we send only its starting time slot number. Node  $v_i$  in uplink virtual group  $V_{c,j}$  will use  $p_i \cdot 2^{j-1}$  time slots starting from its start slot to deliver its  $p_i$  packets. Once the nodes have received their schedules, they start communicating based on Phase II uplink groups and Phase II downlink groups. Subsequently, transmission failures, downlink communications are handled in the same fashion as Phase I but using Phase II grouping. Namely, if a transmission fails the corresponding node will receive a new slot number for retransmission after its group's superframe, and it will retrieve this information from the associated field in the downlink message as given in equation (1).

4) *Latency Analysis:* In this section, we theoretically prove that NOVA is a 2-approximation algorithm for minimizing latency, i.e., the latency achieved through NOVA is at most 2 times the latency that an optimal algorithm will achieve (Theorem 2). We analyze latency performance without considering the time needed for retransmission. Retransmission is needed based on link quality and will be needed equivalently for both an optimal algorithm and our algorithm.

**Theorem 2.** NOVA is a 2-approximation algorithm for minimizing latency.

*Proof:* Let  $V_{c^*, j^*}$  be the virtual group whose superframe finishes last, i.e., determines the latency  $L$  under NOVA. That is,

$$L = \max_{c,j} \{L_{c,j}\} = L_{c^*, j^*} = \sum_{v_i \in V_{c^*, j^*}} p_i \cdot 2^{j^*-1}$$

Let  $v^*$  be the last node that was assigned to a virtual group by NOVA, and let  $p^*$  denote its total number of packets. Since NOVA assigns the nodes in non-increasing order of  $p_i$ ,  $v^*$  must be the node with the smallest number of packets. That is,

$$p^* = \min_i \{p_i\} \quad (5)$$

Being the last node,  $v^*$  was assigned to virtual group  $V_{c^*, j^*}$  and that is why  $V_{c^*, j^*}$  determined  $L$ . Node  $v^*$  was not assigned to any other virtual group because it was resulting in a superframe of length greater than or equal to  $L_{c^*, j^*}$  (i.e.,  $L$ ). Hence, for every virtual group  $V_{c,j}$ , Condition (6) must be true.

$$L \leq L_{c,j} + p^* \cdot 2^{j-1} \quad (6)$$

Since Condition (6) is true for every virtual group, it is also true for the virtual group with the shortest superframe. Let  $V^{\min}$  be the virtual group having the shortest superframe, and let its superframe length be  $L^{\min}$  and its spreading factor be  $j^{\min}$ . We can rewrite Condition (6) for virtual group  $V^{\min}$  as follows.

$$L \leq L^{\min} + p^* \cdot 2^{j^{\min}-1} \quad (7)$$

Based on equation (5), we can write the following inequality:

$$L^{\min} = \sum_{v_i \in V^{\min}} p_i \cdot 2^{j^{\min}-1} \geq p^* \cdot 2^{j^{\min}-1} \quad (8)$$

Therefore, from (7) and (8),

$$L \leq L^{\min} + p^* \cdot 2^{j^{\min}-1} \leq L^{\min} + L^{\min} = 2L^{\min}. \quad (9)$$

$L^{\text{opt}}$  be the latency generated by an optimal subgrouping algorithm. That is,  $L^{\text{opt}}$  is the longest superframe among all subgroups created by an optimal algorithm. And,  $L^{\min}$  is the shortest superframe among all subgroups created by NOVA that assigns each node to the virtual group that minimizes the completion time (latency) at each step. Now we want to determine a lower bound of  $L^{\text{opt}}$  in terms of  $L^{\min}$  as follows. If  $L^{\min} > L^{\text{opt}}$ , this means that some assignment exists where all superframe lengths will be shorter than even the shortest superframe length achieved by NOVA. This contradicts the fact that NOVA considers all possible assignments for each node and chooses the one that minimizes the completion time. Therefore,  $L^{\min} \not> L^{\text{opt}}$ , meaning that  $L^{\min}$  is a lower bound of  $L^{\text{opt}}$ . That is,

$$L^{\min} \leq L^{\text{opt}} \quad (10)$$

Thus, by equation (9),

$$L \leq 2L^{\min} \leq 2L^{\text{opt}}. \quad (11)$$

## V. EXPERIMENTS

First we evaluate the proposed protocol through small-scale physical experiments. Later we complement through large-scale simulations in NS-3.

### A. Experiment Setup

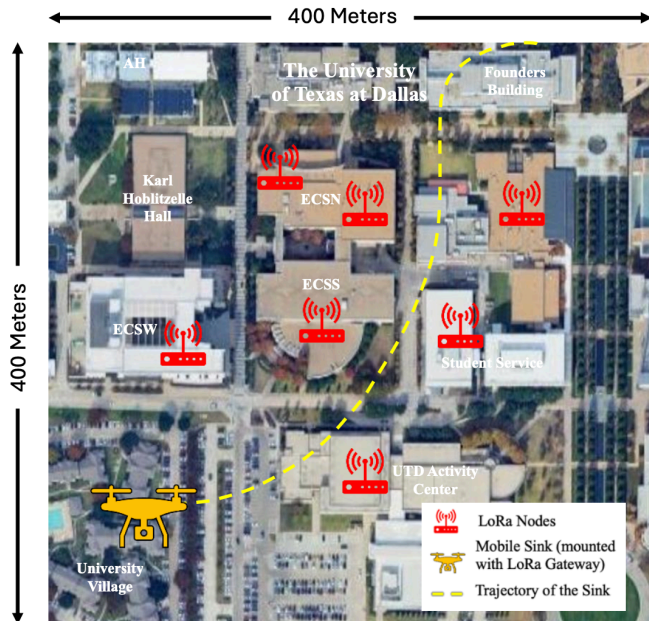


Fig. 3: Outdoor LoRa deployment

We conduct our experiments in an outdoor LoRa nodes deployment on a University of Texas at Dallas campus, covering an area of approximately 400 m  $\times$  400 m, as shown in Fig. 3. The setup consists of seven LoRa nodes, each using a Dragino LA66 LoRaWAN module [41] with an Arduino Uno R3 [42]. A USRP B200 [43] serves as the LoRa gateway. Two physical channels, operating at 903 and 911 MHz, are configured with a bandwidth of 125 kHz and a coding rate of 4/5. The nodes transmit packets using spreading factors 9 and 10, thereby creating  $2 * 2 = 4$  virtual groups. The gateway, mounted on a mobile sink, periodically enters the network coverage area once every hour. Each node generates data according to a Poisson process with an average rate parameter ( $\lambda$ ), leading to a varying number of packets across nodes, thereby emulating heterogeneous traffic in the LoRa network. Namely, each node generates packets following a Poisson distribution with mean  $\lambda$  which is different across nodes. We generate these mean values  $\lambda$  using a normal distribution, where the mean is the packet generation rate. We conduct experiments with different packet generation rates. For each experiment, a different average value  $\lambda$  is assigned to each node. As a result, since the Poisson

mean varies across nodes, each node generates a different number of packets over time. The deployment area is designed such that every node can communicate with the mobile sink from any location using either of the used spreading factors. When the gateway enters the network, our proposed protocol is activated to initiate the data harvesting process.

### B. Baselines and Metrics

To evaluate our proposed approach, we compare it with two baselines: an exhaustive optimal solution and Burst-MAC [9], a TDMA protocol that assigns slots using a hash-based scheme to handle burst traffic under a stationary gateway. Burst-MAC is the closest existing work that can be applied to solve our problem (though not quite effectively) as Burst-MAC performs data collection through virtual grouping for burst traffic. To our knowledge, there is no other existing work that can be used as a baseline. The evaluation is done in terms of the following metrics.

1) *Packet Reception Rate (PRR)*: The packet reception rate (PRR) is defined as the fraction of the total number of packets successfully received at the gateway considering the total number of packets transmitted by all LoRa nodes.

$$\text{Packet Reception Rate} = \frac{\text{Total Received Packets}}{\text{Total Transmitted Packets}} * 100$$

2) *Energy*: We define energy consumption as the total amount of energy expended by a node to successfully transmit an uplink packet to the gateway. For evaluation, we measure the average energy consumption as defined below for each packet transmitted by a node:

$$\text{Energy} = \frac{\text{Total energy for transmission and retransmissions}}{\text{Number of successful packet transmissions}}$$

3) *Network Latency*: We measure the Network Latency ( $L$ ) as in equation (2), i.e., the total time required to deliver all packets to the mobile sink in Phase II, considering only the uplink packets successfully received by the mobile sink.

### C. Results

For this experiment, we vary the packet generation rate ( $\lambda$ ) per node and measure the PRR, energy consumption per packet, and Network Latency. We present the results in Fig. 4. As the packet generation rate increases, the total number of data packets in the LoRa network also increases, resulting in a more bursty traffic pattern.

Fig. 4(a) compares the PRR of our approach with the Optimal Solution and Burst-MAC. Our proposed approach achieves PRR values nearly identical to the Optimal Solution because both follow the same transmission structure, differing only in the scheduling strategy generated before Phase II. On average, both approaches reach a PRR of about 91%, while Burst-MAC attains a lower average PRR of roughly 81%. This degradation in Burst-MAC stems from its inability to manage downlink-uplink collisions: when multiple nodes transmit concurrently, the gateway sends ACKs for packets it successfully receives, and these downlink ACKs can interfere with ongoing uplink transmissions. Without a mechanism to

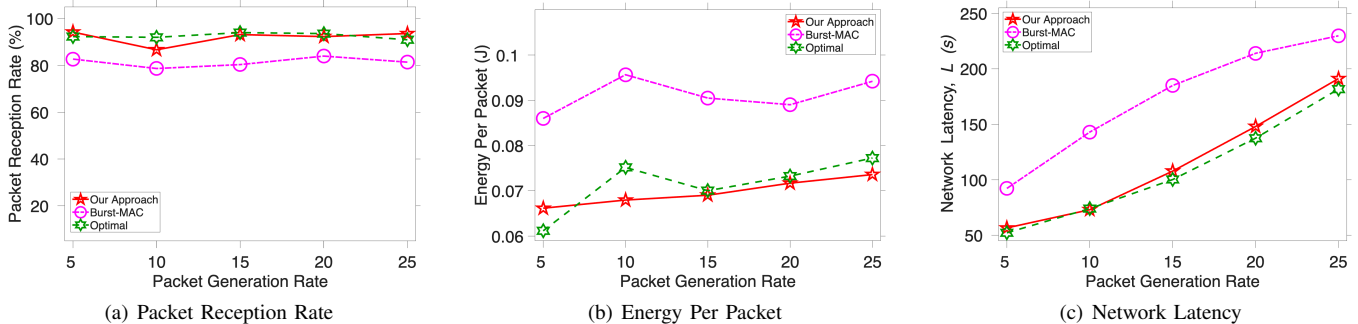


Fig. 4: Experiment results under varying packet generation rate.

mitigate such interference, Burst-MAC suffers more packet loss. In contrast, our approach includes an effective collision-handling mechanism, leading to a PRR approximately 1.13 $\times$  higher than that of Burst-MAC.

Fig. 4(b) compares the energy efficiency of our approach with Burst-MAC. The results show that the average energy consumed per successfully received packet in both our approach and the Optimal Solution increases only slightly as the packet generation rate grows. In contrast, Burst-MAC incurs significantly higher energy consumption because increased traffic leads to longer transmission time and higher network load. As the packet generation rate rises, some packets are not received by the gateway despite being transmitted correctly. Without acknowledgments, nodes trigger retransmissions, further increasing energy usage per successful delivery. Overall, our approach achieves 24% lower energy consumption per successful packet compared to Burst-MAC.

Fig. 4(c) compares the network latency of our approach with the baseline schemes. The Optimal Solution achieves the lowest latency since it constructs Phase II schedules that minimize the longest superframe. Our approach closely follows this behavior, achieving near-optimal latency across all scenarios. As the number of packets increases, latency naturally grows for all schemes; however, the latency under our approach consistently remains close to that of the optimal solution and always within a factor of two. In contrast, Burst-MAC requires more superframes as traffic increases—each node transmits only during its assigned slot—resulting in substantially longer data collection times. Overall, our approach delivers significantly lower latency than Burst-MAC, and achieves near-optimal performance even under high network load.

The performance difference among different techniques were not well-visible due to the small scale of the network in real experiment. Hence, simulation results were provided to observe performance difference in large-scale networks.

## VI. SIMULATION

In this section, we evaluate the performance of our approach through simulations in NS-3, incorporating a gateway mounted on a mobile sink and up to 1000 LoRa nodes. We present

results across a range of scenarios, including variations in network size, data heterogeneity, and virtual groups.

### A. Simulation Setup

We use the LoRaWAN NS-3 module provided in [44]. This module provides classes for modeling the modulation and medium access behavior of LoRaWAN networks, supports large-scale deployments, and allows detailed monitoring of network performance. We consider a deployment area consisting of 5 LoRa networks and a total of 1000 nodes, randomly distributed across 5 networks and positioned within a disc centered at the gateway. A mobile sink, equipped with the gateway, moves along a predefined trajectory. The sink's speed is computed based on the time needed to collect all network data and the length of its traversal path within the LoRa network. The sink enters the deployment area at a fixed interval, once every hour. Upon completing data harvesting from one network, it moves to the next network, continuing this process sequentially. After harvesting data from all the nodes, the mobile sink returns back to the base station. Each node generates packets with a 20-byte payload and a packet generation rate  $\lambda$ , which varies across different scenarios. We consider 4 spreading factors: 7—10. The system operates at a carrier frequency of 915 MHz using nine physical channels, each with a bandwidth of 125 kHz, a coding rate of 4/5, and a transmit power of 14 dBm. We perform the simulation for up to 36 virtual groups.

### B. Baseline and Metrics

We compare our approach against three baselines: an exhaustive optimal solution, Burst-MAC, and LoRaWAN. The evaluation uses the same three performance metrics listed before: PRR, energy consumption per packet, and network latency ( $L$ ). In addition, we evaluate the total time required by the mobile sink during the entire protocol including Phase I, Phase II, and time needed to handle transmission failures, represented as  $(L + \text{Overhead} + \text{Retx})$ , where *overhead* accounts for communication, coordination, and synchronization time, and *Retx* is the time required for retransmissions.

### C. Comparison with Baselines Including the Optimal Solution

We compare our approach with the Optimal Solution and the other baselines in Fig. 5. Since computing the optimal schedule is NP-hard and requires exponential time as the network size grows, we limit the optimal evaluation to networks with up to 25 nodes. The setup uses 2 physical channels and 2 spreading factors, forming 4 virtual groups in total. Each node generates 5 packets per unit time.

As shown in Fig. 5(a), all approaches except LoRaWAN maintain a PRR above 80% as the network size increases. In contrast, LoRaWAN experiences a sharp decline in PRR, dropping to 65% at 25 nodes due to its reliance on the pure ALOHA mechanism, which lacks collision avoidance and leads to a large number of packet collisions.

Fig. 5(b) illustrates the energy consumption per successfully transmitted packet. Both our proposed approach and the Optimal approach exhibit similar and consistent energy consumption. However, Burst-MAC and LoRaWAN consume 1.5x and 1.8x more energy compared to our approach. As the number of nodes increases, energy consumption also rises for LoRaWAN because more packets are added to the network, leading to more collisions. To successfully deliver packets, nodes perform multiple retransmissions, which increases the per-packet energy consumption. In Burst-MAC, the larger number of nodes causes multiple nodes to use the same slot for transmission, resulting in collisions, and consequently requiring retransmissions as well.

As shown in Fig. 5(c), all four approaches initially exhibit similar latency because the small network size results in few collisions and ample virtual channels relative to the number of nodes. As the network grows, however, both Burst-MAC and LoRaWAN experience a sharp increase in latency. In Burst-MAC, each virtual group requires a number of time slots per round equal to its node count, so adding nodes lengthens each round and directly increases total latency. In contrast, our approach maintains latency values close to the Optimal solution by using a virtual-grouping strategy that minimizes the length of the longest superframe. The schedule it produces closely follows the optimal pattern, and the resulting latency consistently stays within a factor of two of the optimal, confirming the approximation bound of 2.

### D. Comparison with Baselines Excluding Optimal Solution

1) *Performance under varying number of nodes:* Figure 6 shows system performance as the network scales from 200 to 1000 nodes, with 20 virtual groups and a packet generation rate of 5 per hour. Across all metrics, our approach consistently outperforms or matches the baselines.

As shown in Figure 6(a), our approach maintains PRR above 94% even at high node counts. Burst-MAC holds around 88%, while LoRaWAN drops sharply to 14% at 1000 nodes due to its pure ALOHA access, which offers no collision avoidance. In small networks (200 nodes), LoRaWAN and Burst-MAC perform similarly, but as density increases, Burst-MAC mitigates collisions better. Our method surpasses both

by dynamically reallocating slots among nodes, reducing collisions and retransmissions. On average, PRR is 2.8x higher than LoRaWAN and 1.18x higher than Burst-MAC.

Figure 6(b) shows that per-packet energy remains stable in our approach, rising slightly from 28 mJ to 29.1 mJ due to lightweight, autonomous scheduling that adjusts node activity. Burst-MAC increases modestly from 40 mJ to 41.1 mJ, whereas LoRaWAN's energy escalates dramatically under dense conditions, reaching 173.6–567.9 mJ, highlighting inefficiency in large networks.

Figure 6(c) illustrates latency trends. Our approach exhibits near-linear growth, reaching 2366 seconds at 1000 nodes—35% lower than Burst-MAC—by balancing slot allocation and leveraging multi-channel and multi-spreading factor diversity to reduce collisions and retransmissions. In contrast, latency for Burst-MAC and LoRaWAN grows exponentially, with LoRaWAN's latency being unmeasurable at 800–1000 nodes due to excessive collisions.

2) *Performance under varying packet generation rate:* In this simulation, the packet generation rate varies from 1 to 10 packets per hour, with 1000 nodes and 20 virtual groups. Figure 7 compares performance across the evaluated schemes.

As shown in Figure 7(a), PRR decreases for all approaches as traffic increases. Our approach maintains a high PRR of around 94%, while Burst-MAC and LoRaWAN achieve roughly 86% and 15%, respectively. This is due to our dynamic virtual group mechanism, which localizes collisions within small groups, preserving reliability even under heavy traffic. LoRaWAN, in contrast, suffers from widespread collisions from its pure ALOHA access, causing rapid PRR degradation. Overall, our approach outperforms Burst-MAC and LoRaWAN by 1.1x and 6.1x, demonstrating robustness and scalability under bursty traffic.

Figure 7(b) shows per-packet energy consumption. Our approach and Burst-MAC remain stable due to their TDMA-based structures, but our method is more efficient: energy drops slightly from 29.1 mJ to 27.5 mJ with higher packet rates thanks to adaptive scheduling that reduces idle listening and retransmissions. Burst-MAC consumes 39–41 mJ per packet, while LoRaWAN incurs 442–813 mJ due to frequent retransmissions and channel contention.

Figure 7(c) illustrates latency. Our approach achieves the lowest latency (2501 s) compared to Burst-MAC (6775 s), a 37% improvement. LoRaWAN's latency remains unassessed at high packet rates as collisions prevent packet delivery.

3) *Performance under varying virtual groups:* In this simulation, we vary the number of virtual groups for uplink transmission while keeping the total number of LoRa nodes fixed at 1000 and the packet generation rate at 5 packets per hour. Figure 8 compares performance across different metrics.

As shown in Figure 8(a), increasing virtual groups gradually reduces the PRR for both our approach and Burst-MAC. In our approach, PRR drops to about 77% at 36 virtual groups due to more concurrent transmissions exceeding the gateway's processing and acknowledgment capacity, leading to packet loss. LoRaWAN shows a slight PRR improvement from

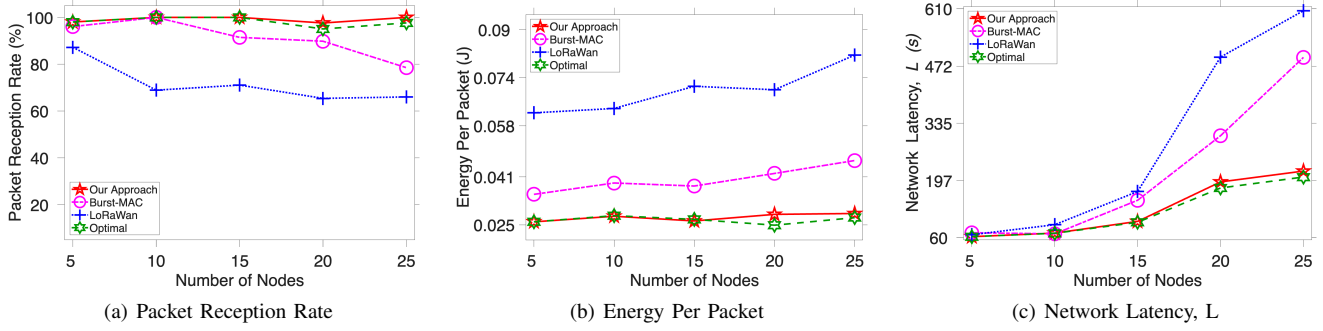


Fig. 5: Optimal results under varying number of nodes (Simulation).

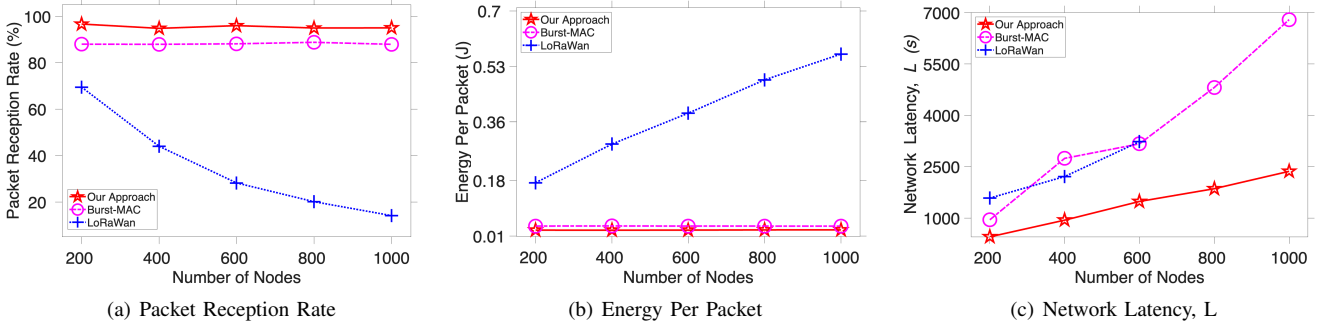


Fig. 6: Simulation results under varying number of nodes.

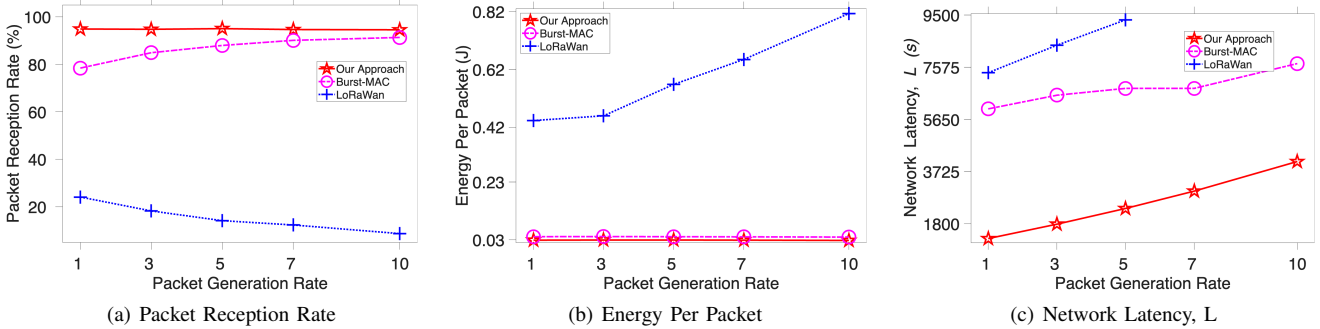


Fig. 7: Simulation results under varying packet generation rate.

frequency diversity but remains far below our scheme. Overall, our approach achieves 4.5 $\times$  higher PRR than LoRaWAN and 1.1 $\times$  higher than Burst-MAC, highlighting its superior ability to manage channel diversity and minimize collisions.

Figure 8(b) shows per-packet energy consumption. Our approach consistently uses the least energy, averaging 29.1 mJ and rising slightly to 32.1 mJ due to occasional retransmissions. Burst-MAC remains around 42 mJ, while LoRaWAN consumes 435 mJ per packet, decreasing slightly with more virtual groups. Our method achieves nearly 96% energy savings over LoRaWAN and 28% over Burst-MAC.

Figure 8(c) presents latency results. All protocols show reduced latency. Here both the number of nodes and the packet

generation rate are fixed, while the number of virtual groups changes. As the number of virtual groups increases, more nodes can transmit in parallel. This reduces the total time required to collect all data. With fewer virtual groups, more nodes share the same group, causing higher collision rates. LoRaWAN is particularly affected under this scenario at 4 virtual groups; collisions are so severe that the latency exceeds the plottable range. Our approach further outperforms Burst-MAC by 36% due to adaptive slot scheduling.

4)  $(L+Overhead+ReTx)$  under varying network parameters: We compare the total time ( $L+Overhead+ReTx$ ) with baseline approaches in Fig. 9. In Fig. 9(a), with a fixed packet generation rate of 5 and 12 virtual groups, our ap-

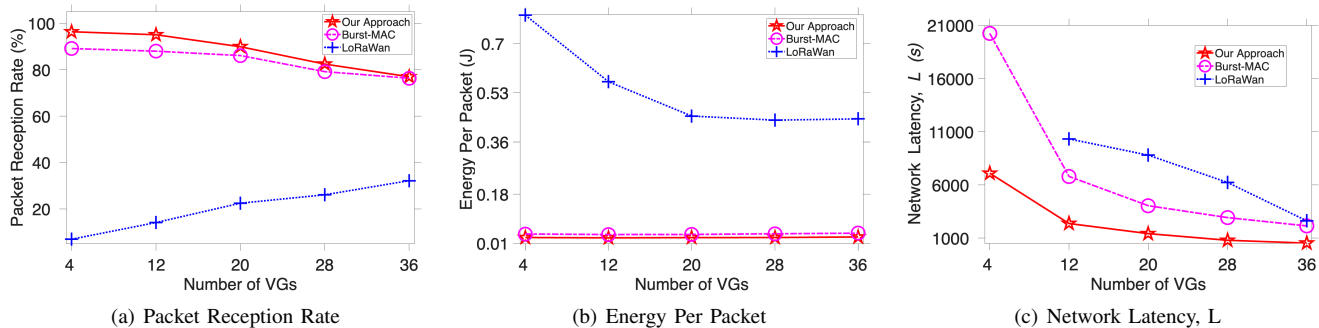


Fig. 8: Simulation results under varying number of virtual groups.

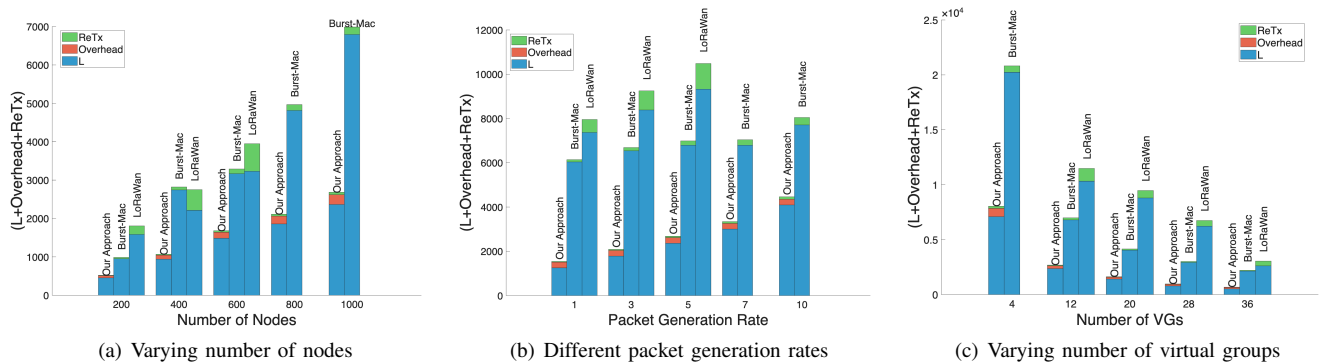


Fig. 9: Simulation results under varying number of virtual groups.

proach achieves lower total collection time than others, despite including overhead. This improvement stems from efficient scheduling that reduces network latency and ensures collision-free transmissions in both phases. Burst-MAC and LoRaWAN require 2.3 $\times$  and 2.8 $\times$  more time, respectively.

Fig. 9(b) evaluates the effect of increasing packet generation rates with 1000 nodes and 12 groups. In our approach, Phase I overhead remains constant, as the number of nodes and virtual groups remain fixed. For the baselines, total collection time increases with higher packet rates because more packets must be transmitted. Unsynchronized schemes like LoRaWAN fail to achieve 80% data collection at higher rates like 7 and 10, due to random transmissions and long delays, and are hence omitted. On average, Burst-MAC and LoRaWAN require 2.7 $\times$  and 4.5 $\times$  time, respectively, compared to our approach.

In Fig. 9(c), we vary the number of virtual groups with 1000 nodes and a packet generation rate of 5. Larger virtual group sizes reduce total collection time by enabling more parallel, collision-free transmissions. For a group size of 4, overhead is slightly higher, as fewer nodes transmit simultaneously. Across all group sizes, Burst-MAC and LoRaWAN require 2.8 $\times$  and 2.1 $\times$  more time, respectively. Overall, our approach outperforms baselines by efficiently scheduling transmissions and minimizing network latency, resulting in significantly lower total collection time.

## VII. CONCLUSION

We have proposed a novel data harvesting protocol that achieves near-optimal latency in remote LoRa deployments using a mobile sink. Optimizing latency for data harvesting in a LoRa network is a challenging problem that has not been studied before. We have formulated the latency minimization problem as an optimization problem, proved its NP-hardness, and proposed a novel 2-approximation algorithm, which the protocol adopts on the fly for conflict-free transmission scheduling. This work presents the first provable constant approximation bound on latency for a LoRa network. We have evaluated the protocol through both physical experiments and NS-3 simulations. Experiments show that the latency achieved through our approach can be as close as 1.07 times the optimal latency. Simulations demonstrate that our approach reduces latency by up to 80% compared to a state-of-the-art approach, while also decreasing energy consumption by 30%. In the future, we plan to evaluate the protocol using a large-scale LoRa network deployment.

## ACKNOWLEDGEMENT

The work was supported by the US National Science Foundation through grants CNS-2601685 and CNS-2602744, and by the US Office of Naval Research through grant N00014-23-1-2151.

## REFERENCES

- [1] “LoRaWAN,” <https://www.lora-alliance.org>.
- [2] J. Petajajarvi, K. Mikhaylov, R. Yasmin, M. Hämäläinen, and J. Iinatti, “Evaluation of lora lpwan technology for indoor remote health and wellbeing monitoring,” *International Journal of Wireless Information Networks*, vol. 24, 06 2017.
- [3] Y. Li, Z. Wang, and Y. Song, “Wireless sensor network design for wildfire monitoring,” in *2006 6th World Congress on Intelligent Control and Automation*, vol. 1, 2006, pp. 109–113.
- [4] L. Li, J. Ren, and Q. Zhu, “On the application of lora lpwan technology in sailing monitoring system,” *2017 13th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, pp. 77–80, 2017. [Online]. Available: <https://api.semanticscholar.org/CorpusID:16850262>
- [5] Business Insider, “Smart farming in 2020: How iot sensors are creating a more efficient precision agriculture industry,” <https://www.businessinsider.com/smart-farming-iot-agriculture>, 2020.
- [6] S. Benaissa, P. A. David, E. Tanghe, J. Trogh, L. Martens, L. Vandaele, L. Verloock, F. Tuytens, B. Sonck, and W. Joseph, “Internet of animals: characterisation of lora sub-ghz off-body wireless channel in dairy barns,” *Electronics Letters*, vol. 53, pp. 1281–1283, 2017.
- [7] J. P. Bardyn, T. Melly, O. Seller, and N. Sornin, “Iot: The era of lpwan is starting now,” in *ESSCIRC Conference 2016: 42nd European Solid-State Circuits Conference*, 2016, pp. 25–30.
- [8] <https://www.i-scoop.eu/internet-of-things-guide/iot-network-lora-lorawan/>.
- [9] A. Jain, M. A. Haque, A. Saifullah, and H. Zhang, “Burst-mac: A mac protocol for handling burst traffic in lora network,” in *2024 IEEE Real-Time Systems Symposium (RTSS)*, 2024, pp. 148–160.
- [10] 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.
- [11] 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.
- [12] S. Fahmida, V. P. Modekurthy, D. Ismail, A. Jain, and A. Saifullah, “Real-time communication over lora networks,” in *2022 IEEE/ACM Seventh International Conference on Internet-of-Things Design and Implementation (IoTDI)*, 2022, pp. 14–27.
- [13] S. Fahmida, A. Jain, V. P. Modekurthy, D. Ismail, and A. Saifullah, “Rtpl: A real-time communication protocol for lora network,” *ACM Trans. Embed. Comput. Syst.*, vol. 24, no. 1, Dec. 2024. [Online]. Available: <https://doi.org/10.1145/3702209>
- [14] 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.
- [15] A. Jain, P. Modekurthy, and A. Saifullah, “Control over low-power wide-area networks,” in *2024 ACM/IEEE 15th International Conference on Cyber-Physical Systems (ICCPs)*, 2024, pp. 192–201.
- [16] 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.
- [17] 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.
- [18] P. Marcellis, V. S. Rao, and R. V. Prasad, “DaRe: Data recovery through application layer coding for loRaNANs,” *IoTDI '17*, 2017.
- [19] 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 *ACM/IEEE IPSN '18*.
- [20] R. Eletreby, D. Zhang, S. Kumar, and O. Yağan, “Empowering low-power wide area networks in urban settings,” in *SIGCOMM*, 2017, pp. 309–321.
- [21] C. Li, H. Guo, S. Tong, X. Zeng, Z. Cao, M. Zhang, Q. Yan, L. Xiao, J. Wang, and Y. Liu, “Nelora: Towards ultra-low snr lora communication with neural-enhanced demodulation,” in *SenSys*, 2021, pp. 56–68.
- [22] 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.
- [23] X. Xia, Y. Zheng, and T. Gu, “Ftrack: Parallel decoding for lora transmissions,” in *SenSys*, 2019, pp. 192–204.
- [24] 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.
- [25] X. Wang, L. Kong, L. He, and G. Chen, “mlora: A multi-packet reception protocol in lora networks,” in *ICNP*. IEEE, 2019, pp. 1–11.
- [26] S. Tong, Z. Xu, and J. Wang, “Colora: Enabling multi-packet reception in lora,” in *INFOCOM*. IEEE, 2020, pp. 2303–2311.
- [27] 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.
- [28] M. A. Haque and A. Saifullah, “Handling jamming attacks in a LoRa network,” in *2024 IEEE/ACM Ninth International Conference on Internet-of-Things Design and Implementation (IoTDI)*, 2024, pp. 146–157.
- [29] B. Hu, Z. Yin, S. Wang, Z. Xu, and T. He, “Sclora: Leveraging multi-dimensionality in decoding collided lora transmissions,” in *2020 IEEE 28th International Conference on Network Protocols (ICNP)*, Oct 2020, pp. 1–11.
- [30] A. Gamage, J. C. Liando, C. Gu, R. Tan, and M. Li, “Lmac: efficient carrier-sense multiple access for lora,” in *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '20. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: <https://doi.org/10.1145/3372224.3419200>
- [31] C. Pham, A. Bounceur, L. Clavier, U. Noreen, and M. Ehsan, “Investigating and experimenting interference mitigation by capture effect in lora networks,” in *Proceedings of the 3rd International Conference on Future Networks and Distributed Systems*. ACM, 2019, p. 31.
- [32] J. Ortín, M. Cesana, and A. Redondi, “Augmenting lorawan performance with listen before talk,” *IEEE Transactions on Wireless Communications*, vol. 18, no. 6, pp. 3113–3128, 2019.
- [33] —, “How do aloha and listen before talk coexist in lorawan?” in *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2018, pp. 1–7.
- [34] H. Cho and S. W. Kim, “An anti-collision algorithm for localization of multiple chirp-spread-spectrum nodes,” *Expert Syst. Appl.*, vol. 39, no. 10, p. 8690–8697, aug 2012. [Online]. Available: <https://doi.org/10.1016/j.eswa.2012.01.213>
- [35] L.-H. Shen, C.-H. Wu, W.-C. Su, and K.-T. Feng, “Analysis and implementation for traffic-aware channel assignment and contention scheme in lora-based iot networks,” *IEEE Internet of Things Journal*, vol. 8, no. 14, pp. 11 368–11 383, 2021.
- [36] N. El Rachkidy, A. Guitton, and M. Kaneko, “Collision resolution protocol for delay and energy efficient lora networks,” *IEEE Transactions on Green Communications and Networking*, vol. 3, no. 2, pp. 535–551, 2019.
- [37] R. Piyare, A. L. Murphy, M. Magno, and L. Benini, “On-demand lora: Asynchronous tdma for energy efficient and low latency communication in iot,” *Sensors*, vol. 18, no. 11, 2018. [Online]. Available: <https://www.mdpi.com/1424-8220/18/11/3718>
- [38] C. Gu, R. Tan, and X. Lou, “One-hop out-of-band control planes for multi-hop wireless sensor networks,” *ACM Trans. Sen. Netw.*, vol. 15, no. 4, jul 2019. [Online]. Available: <https://doi.org/10.1145/3342100>
- [39] M. Rizzi, P. Ferrari, A. Flammini, E. Sisinni, and M. Gidlund, “Using lora for industrial wireless networks,” in *2017 IEEE 13th International Workshop on Factory Communication Systems (WFCS)*, 2017, pp. 1–4.
- [40] J. Rao and S. Biswas, “Data harvesting in sensor networks using mobile sinks,” *IEEE Wireless Communications*, vol. 15, no. 6, pp. 63–70, 2008.
- [41] “Dragino la66 lorawan module,” <https://www.dragino.com/products/lora/item/230-la66-lorawan-module.html>.
- [42] “Arduino uno rev3,” <https://store-usa.arduino.cc/products/arduino-uno-rev3>.
- [43] “Ettus research,” <https://www.ettus.com/product/>.
- [44] 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.