RALoRa: Rateless-Enabled Link Adaptation for LoRa Networking

Kang Yang, Miaomiao Liu, and Wan Du, *Member, IEEE*

***Abstract*— Both our experiments and previous studies show that LoRa links vary dynamically, which makes data transmission unreliable and consumes much energy of sensor nodes by retransmissions. This paper presents *RALoRa*, a Rateless-enabled link Adaptation system for LoRa networks. Rateless coding approaches the optimal data rate of a link by continuously transmitting encoded data with an initial data rate. However, LoRa's modulation, Chirp Spread Spectrum (CSS), introduces unique challenges to rateless-enabled transmissions. With CSS, Spreading Factor (SF) simultaneously determines the initial data rate and the concurrent transmissions of multiple links. This dual role of SF requires the co-design of coding and networking. We thus formulate an optimization problem for allocating network resources (SF, frequency channels, and transmission power) and coding parameters (block size and packet size) to all sensor nodes. A key component of this formulation is a rateless-aware network model that estimates the data transmission results of sensor nodes based on their link quality and transmission setting. Given that the optimization problem for obtaining the best transmission setting of all sensor nodes is NP-complete, a two-stage heuristic algorithm is designed. A Kalman filter-based link quality predictor is developed to capture the link quality variation. We implement *RALoRa* on commodity LoRa hardware. Extensive experiments on a real testbed show that *RALoRa* extends the lifetime of LoRaWAN by 66.1 %.**

***Index Terms*— Low-power wide-area networks, LoRaWAN, rateless coding, data rate adaptation.**

I. Introduction

THE Long Range (LoRa) networks, increasingly deployed in environmental monitoring applications, offer the advantages of long-range communication and low-power consumption, making them well-suited for data collection in remote or wild areas [1], [2], [3]. In such applications, where battery-powered nodes are often spread across vast areas, the reliability and energy efficiency of data collection become paramount. Our experiments and previous studies [4], [5] have shown that the quality of LoRa links is subject to dynamic variations. Consequently, Adaptive Data Rate (ADR) is crucial to approach the optimal data rate of links, achieving energy efficiency and reliability in data transmission [6].

Current LoRa networks use the ADR algorithm specified in LoRaWAN (Long Range Wide Area Network) [7]. Nodes can select from four data rates: 5.47, 3.13, 1.76, and 0.98 kb/s, each associated with a specific Signal-to-Noise Ratio (SNR) limit, namely −7.5, −10.0, −12.5, and −15.0 dB, respectively. When the SNR of a received packet exceeds the corresponding SNR limit, the likelihood of successful packet demodulation is high. Conversely, if the SNR falls below this threshold, there is a high probability of demodulation failure. To transmit a packet, a node first predicts the SNR of its upcoming transmission using the SNRs of multiple previously-transmitted packets and selects a data rate whose SNR limit is slightly lower than the predicted SNR. However, this ADR algorithm has three limitations. *1) The gap between two adjacent data rates is large.* For a link that can support a rate of 5.0 kb/s, setting the data rate to 5.47 kb/s can lead to an increase in transmission failures. Conversely, choosing a rate of 3.13 kb/s nearly doubles the transmission time. *2) It sets the data rate for each individual link without considering the other links in the network.* LoRa employs Chirp Spread Spectrum (CSS) modulation [6], [7], where data rate is determined by Spreading Factor (SF). Links with differing SFs can transmit concurrently on the same channel, whereas links with the same SF may interfere with each other. Hence, a node needs to consider its peers using an SF before shifting to that SF. *3) Given the low duty cycle of LoRa networks [8], predicting link quality using recently received packets is challenging.* For instance, if the transmission cycle is 15 minutes, the third most recently received packet would have arrived 30 minutes ago, which is insufficient for providing meaningful link quality information for the next cycle.

This paper leverages rateless coding to approach the optimal data rate of a LoRa link. Rateless coding continuously transmits encoded bits until the receiver successfully recover the original data [9]. The achieved data rate can be calculated by the ratio between the total number of transmitted encoded bits over the total transmission time. However, these bit-level rateless codes has high computational complexity, which are not suitable for low-cost half-duplex LoRa nodes. To identify an effective rateless code, we first examine LoRa links on a testbed of ten nodes (details in Section III-B1). Our experimental results reveal: 1) When the initial data rate is set high (small SF), many received packets are corrupted, but only a few error bits are found in these packets. 2) Error bits tend to be clustered, suggesting that when a packet is divided into multiple

Manuscript received 26 September 2023; revised 21 February 2024; accepted 5 April 2024; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor J.-W. Lee. This work was supported in part by the Financial Assistance award from the Economic Development Administration, Farms Food Future; in part by NSF under Grant 2239458 and Grant 2008837; and in part by University of California (UC) Merced Spring 2023 Climate Action Seed Competition Award. *(Corresponding author: Wan Du.)*

Kang Yang and Wan Du are with the Department of Computer Science and Engineering, University of California, Merced, CA 95340 USA (e-mail: kyang73@ucmerced.edu).

Miaomiao Liu was with the Department of Computer Science and Engineering, University of California, Merced, CA 95340 USA. She is now with Visa Inc., Foster City, CA 94404 USA (e-mail: mliu71@ucmerced.edu).

Digital Object Identifier 10.1109/TNET.2024.3392342

blocks, most of these blocks are received correctly. *These observations motivate us to implement block-level rateless coding for LoRa links.*

Block-level rateless coding has been employed in wireless sensor networks for error correction [10]. A node partitions its sensing data into blocks and encodes them into encoded blocks using a lightweight rateless code, Luby Transform (LT) codes [11], [12]. Each encoded block includes a short Cyclic Redundancy Check (CRC) that allows the receiver to verify its correctness. Ideally, the node transmits encoded blocks until the receiver decodes the original data, sending an acknowledgment (ACK) to halt the transmission. Due to the limitations of half-duplex radio, where nodes can't receive ACKs during transmission, Hybrid Automatic Repeat Request (HARQ) is utilized. Nodes first transmit a packet composed of several encoded blocks. If the receiver fails to recover the original data, it sends a negative acknowledgment (NAK), prompting the sender to transmit a second packet with additional blocks. Correctly received blocks from both packets are then used for rateless decoding. Waiting for NAKs and sending extra packets prolong the overall transmission time, deviating from the optimal data rate. To mitigate this, the first packet should contain an appropriate number of encoded blocks with a proper block size to achieve successful decoding. A recent work, eLoRa [13], explores the configuration of block-level rateless coding for LoRa links. It adopts a LoRa link model to set the block size and SF for a link, based on the SNRs of recently received packets. However, eLoRa does not address the three limitations of the LoRaWAN ADR algorithm. 1) It fails to approach the optimal data rate as it uses a fixed packet size, leading to frequent waiting for ACKs and retransmissions of more blocks. 2) It sets the SF of one link without considering its impact on the other links in the network. 3) It simply predicts the SNR of the upcoming transmission based on recently received packets. Our experiments show an average relative error of 82.4 % in SNR prediction.

In this paper, we introduce *RALoRa*, a novel networking solution for data rate adaptation in LoRa networks. We formulate an optimization problem for network resource allocation, including SF, frequency channels, transmission power, and rateless coding parameters (block size and packet size), aiming to maximize the overall lifetime of all nodes. To analyze the problem, we craft a rateless-aware network model designed to estimate the data transmission results of all nodes given their resource setting and link quality. This model adopts rateless-enabled links and bit-level network modeling. First, we package a proper number of rateless-encoded blocks within a single packet by a predicted Block Reception Ratio (BLRR). This block-level rateless link optimization allows nodes to transmit packets at higher data rates, and approach the optimal data rate (*addressing Limitation #1*). Second, to predict BLRR in the context of the entire network, our bit-level network modeling module quantifies the bit error rate of a link by considering resource settings of all nodes, in-network interference among nodes, and the ALOHA-based LoRaWAN MAC (Multiple Access Control) protocol (*addressing Limitation #2*). Specifically, a path loss propagation model is employed to calculate the received signal power at each node, given the transmission power, communication distance, and Path Loss Exponent (PLE). Then, by modeling the ALOHA-based MAC protocol as a Poisson process, the expected interference power is calculated, which represents the signal power from interfering nodes using the same channel and SF.

Having proven the NP-completeness of the above optimization problem, we devise a two-stage heuristic algorithm that includes an offline initialization stage and an online updating stage. The offline stage sets initial parameters for all nodes pre-deployment, optimizing each node's settings to maximize total network lifetime, considering the settings of others. Iterations continue until the lifetime improvement between successive iterations is below a set threshold.

Upon deploying the network in field, the server will run in-situ updating algorithm to adapt nodes' settings to fluctuating link quality at each sensing cycle. After receipt of a packet from a node, we execute a link quality predictor to estimate the PLE for next cycle, leading to an updated rateless-aware network model. With this updated network model, we traverse the node's settings in search of the new settings that maximizes the total lifetime. These revised settings are relayed back to the node via ACK, which will then be employed for data transmission in next sensing cycle.

Existing link quality prediction methods [10], [13] estimate SNR or BER (Bit Error Ratio) by weighted averaging multiple recently received packets' SNRs or BERs. However, these methods are unsuitable for LoRa networks due to their low duty cycle. Furthermore, in LoRa networks, neither SNR nor BER can reliably indicate link quality. Given CSS modulation, SNR and BER are influenced not only by the strength of the received signal but also by interference from other nodes using the same channel and SF. Even with identical received signal strength, different SFs would yield different BERs and SNRs.

We adopt Kalman Filter (KF) [14] to predict PLE for the next cycle using only one packet received in the current cycle (*addressing Limitation #3*). The PLE serves as an indicator of link quality as it is determined solely by the received signal strength, independent of in-network interference. Upon receiving a packet, we calculate the PLE using the measured SNR and BER by inversely solving our network model, which then serves as input to the KF predictor. Our predictor estimates PLE for the next cycle, which is subsequently utilized to compute BER via our network model. Experimental results demonstrate that our PLE predictor can achieve a median BER prediction of 13.3 % when the sensing cycle is 15 minutes.

RALoRa is implemented on commodity LoRa nodes and gateways. In a residential area, we deploy ten nodes and a gateway, with each node transmitting 32-byte sensing data every 15 minutes. To simulate a 450-node network, we reduce the transmission interval to 20 seconds per node [15], resulting in 45 packets sent per node in a 15-minute cycle. Maintaining a 15-minute sensing cycle, our in-situ updating uses the first packet to predict the BER for the 46th packet. Experimental results show that *RALoRa* extends the lifetime of LoRaWAN and eLoRa by 66.1% and 30.4%, respectively.

In summary, this paper makes the following contributions:

- We conduct a series of in-field measurements that suggest block-level rateless coding for LoRa networking.

- We formulate an optimization problem that jointly configures network resources and rateless parameters for reliable and energy-efficient LoRa networking.
- A rateless-aware LoRa network model and a two-stage heuristic algorithm are developed to solve the optimization problem and to adapt to the varying link quality.
- Extensive experiments on a real testbed and a large-scale simulated network show *RALoRa*'s effectiveness.

II. Related Work

Rateless Coding for LoRa: Recent advancements, eLoRa, implemented block-level rateless coding to enhance transmission efficiency [13]. It utilizes a link model to determine the block size and SF for each link. However, *RALoRa* diverges from eLoRa in three key aspects. First, while eLoRa uses a fixed packet size with a constant number of encoded blocks, leading to increased retransmissions and overlooking the overheads of rateless transmissions, *RALoRa* optimizes packet composition. We incorporate an appropriate number of blocks per packet to minimize retransmissions and selectively deactivates rateless coding when the overhead outweighs the benefits. Second, eLoRa determines the SF for a link does not consider the network-wide impact, focusing solely on individual link efficiency. *RALoRa*, in contrast, adopts a holistic network model that accounts for the effect of SF settings on concurrent transmissions and overall network performance. This is particularly vital in LoRa networks, where SF settings play a critical role in data rate and transmission concurrency. The optimization problem and heuristic algorithm presented in this paper are designed to address link adaptation across the entire network. Third, eLoRa modifies node configurations based on the average SNR of several previously received packets, which may not provide accurate predictions due to long sensing cycles. *RALoRa*, on the other hand, employs a Kalman Filter to predict a link's PLE using the SNR and BER from a single, most recent packet. This approach ensures more timely and accurate adjustments to link settings.

Network Modeling for LoRa: Several network models have been proposed to analyze data transmissions and energy consumption in LoRa networks [16], [17], [18]. For instance, EF-LoRa [16] uses a network model for energy fairness, minimizing the energy consumption disparity among nodes. However, these models mainly operate at the packet level and do not incorporate rateless transmissions. *RALoRa*, in contrast, introduces a network model that not only captures BER of links considering inter-node interference but also accommodates the variable packet sizes characteristic of rateless transmissions. Importantly, this model integrates the joint optimization of rateless coding and network resource allocation. *RALoRa*, however, optimizes energy efficiency for traditional LoRa networks where nodes don't communicate with each other.

Forward Error Correction (FEC) for LoRa: LoRa uses Hamming codes [19] with a code rate of 4/5 [6], [20]. Every four bits are attached with a one-bit parity check that can detect a one-bit error in the five bits but cannot correct any error [5]. FLoRa [21] employs packet-level rateless coding for downlink FEC, facilitating firmware updates. These methods cannot leverage the clean bits in the corrupted packets or adapt to link quality variation. Yu et al. [22] use rateless polar codes [23] that enables variable coding rates to reduce the number of packet retransmissions. However, it is not suitable for LoRa networks because the encoder of polar codes requires the specific hardware circuits [24], which is not available in the commodity LoRa devices.

AdapLoRa [18] integrates Reed-Solomon (RS) codes into a symbol-level network model for error correction in LoRa networks. However, RS codes are less suited for LoRa networks due to their inability to effectively handle the bursty error patterns often encountered in LoRa links and the stringent requirements for precise link quality prediction. In contrast, *RALoRa* adopts lightweight rateless coding, offering efficient handling of bursty errors, adaptability to changing link conditions without stringent quality predictions. Furthermore, our network model incorporates rateless-enabled HARQ mechanism, eliminating the need for re-transmitting entire packets. We also developed an innovative Kalman filter-based link quality predictor, further boosting *RALoRa*'s effectiveness in dynamic LoRa network environments.

There are methods [5], [6] that correct bit errors via physical layer information. For example, LLDPC [6] implements a Low-Density Parity Check (LDPC) code for uplink error correction in LoRa. However, these approaches require software defined radios (SDRs) for decoding. *RALoRa* is a link adaptation solution on commodity LoRa hardware, which is orthogonal to these approaches, *i.e.*, we can implement our MAC-level rateless coding on the top of them.

LoRa Throughput: A myriad of studies [25], [26], [27] have targeted packet collisions in LoRa networks to enhance the throughput. For instance, Choir [25] leverages the inherent hardware offsets of the LoRa nodes to decode collided signals. These approaches require SDRs to extract the low-level physical layer information. *RALoRa* is a software solution that can be implemented on top of them.

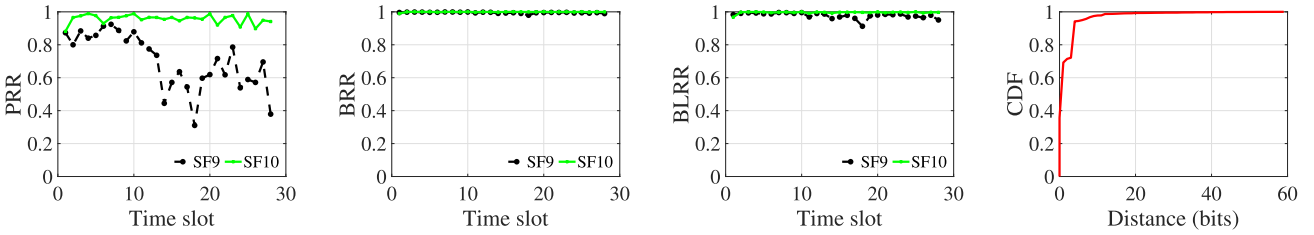
III. Background and Motivation

We begin with an introduction to LoRaWAN, followed by three preliminary experiments designed to motivate this work.

A. LoRa and LoRaWAN

LoRa specifies the CSS modulation for the physical layer and addresses bit errors through FEC. LoRaWAN [7] is a MAC layer protocol that is built on top of the LoRa physical layer. It implements an Adaptive Data Rate (ADR) algorithm to determine the data rate for each node and uses ALOHA to multiplex transmissions from multiple nodes.

LoRaWAN MAC Layer: LoRaWAN operates under three classes. Our focus lies on Class A, which is widely used owing to its low power consumption [7], [26]. Class A nodes empower bidirectional communication where each node's uplink transmission is followed by two short downlink receiving windows. Nodes' transmission time slots are scheduled according to their sensing cycles, with a small random time allocated for ALOHA-based MAC transmissions [7]. We do not consider Class B and C. Class B is activated only when gateways need to periodically send



(a) Packet level (b) Bit level (c) Block level (d) Distance between two error bits

Fig. 1. Packet Reception Ratio (PRR), Bit Reception Ratio (BRR), and Block Reception Ratio (BLRR) of links in our testbed.

downlink messages, and Class C is rarely used due to its high energy consumption [7], [26].

ADR Algorithm in LoRaWAN: According to LoRaWAN specifications, every data rate has an SNR limit [7]. The ADR algorithm estimates a link's SNR by averaging the SNRs of the recent received packets. It then chooses the highest data rate whose SNR limit is lower than the estimated SNR.

B. Motivating Experiments

We delve into the characteristics of bit errors in LoRa links, highlighting the potential of block-level rateless coding. Next, we explore the dynamic link quality, emphasizing the indispensable role of the ADR algorithm. Finally, by examining the effects of in-network interference on BER, we are driven to account for interference when designing ADR algorithms.

1) Block-Level Rateless Coding: We investigate the characteristics of bit errors in LoRa communication links through a series of in-field experiments using a LoRa testbed, detailed in Section V-A. Two LoRa nodes are placed at each location, each transmitting over a different channel and utilizing adjacent SFs to explore the performance of two data rates over independent links. Nodes are deployed across ten different locations, with the average distance from these nodes to the gateway being approximately 308.2 meters. Each node is statically positioned on the ground, while the gateway is stationed on the second floor of a building, near a window, and connected to the Internet via WiFi. Operating on 904.3 and 904.5 MHz channels to minimize interference, each node transmits 32-byte packets at 20-second intervals. The experiment, lasting three hours, focuses on measuring three metrics every six minutes: Packet Reception Ratio (PRR), Bit Reception Ratio (BRR), and Block Reception Ratio (BLRR) [10].

Figure 1(a-c) shows the results for one link, with other links demonstrating similar phenomena. Figure 1(a) reveals that the PRR of SF10 is high (>90 %), while the PRR of SF9 is low (57.4 %). As the data rate increases from 0.98 kb/s (SF10) to 1.76 kb/s (SF9), BER also increases, leading to many packet transmission failures. However, Figure 1(b) indicates that the BRR of SF9 is high. The high BRR and low PRR of SF9 suggest that the number of error bits in corrupted packets is small, although many packets are corrupted. In Figure 1(c), we split each received packet into multiple blocks, each of equal size (4 bytes), and then compute the BLRR. The results reveal that the BLRR of SF9 remains comparable with that of SF10. This is attributed to the burst of error bits, which tend to be concentrated in a few blocks, leaving most blocks error-free. These clean blocks can be exploited by rateless coding.

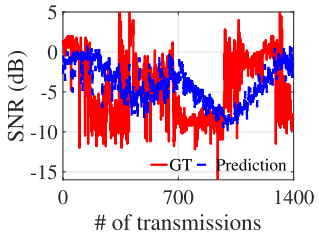


Fig. 2. Illustration of the fluctuating LoRa link quality over time.

Figure 1(d) further investigates the burst phenomenon of error bits for all links. We calculate the distance between two error bits in corrupted packets. If two error bits are adjacent, their distance is 0. We plot the Cumulative Distribution Function (CDF) of the distances between two error bits in all the received packets. For more than 96.8 % of pairs of error bits, their distances are less than 8 bits, and 35.6 % of error bits are adjacent to each other (the distance is 0). Consequently, rateless coding offers the potential to transmit packets using smaller SFs by effectively utilizing clean blocks, which in turn reduces the packet transmission time.

2) Link Quality Variation: Upon analyzing the link quality, with a focus on the SNR of received packets, we observe notable fluctuations over time, as depicted in Figure 2. Our analysis is concentrated on packets from a specific location shown in Figure 1. The SNR varies significantly within a range of -15 dB to 5 dB, posing challenges for accurate SNR prediction. This level of variability is typical in residential areas, where link obstructions by multiple vehicles can drastically reduce SNR. However, the removal of such obstructions, establishing a Line-of-Sight (LoS) path between the node and gateway, typically results in an increase in SNR. A heuristic method is applied, involving a weighted combination of the SNRs of the most recently received packets, to predict the SNR. However, this approach results in an average relative error of 82.4 % in SNR prediction. Specifically, this method incorporates a weighted combination of SNRs from the three latest packets, organized by their reception time and grouped in sets of four for training and testing purposes, following a 7:3 ratio. The training dataset facilitates the optimization of weight values through a brute-force search. This significant inaccuracy in SNR prediction highlights the need for developing a more robust link quality predictor.

3) In-Network Interference: We study the impact of in-network interference on the BER in scenarios with and without interference from other nodes. We set up two scenarios over a duration of one hour: the first involves a LoRa node transmitting packets every 20 seconds on channel CH1 with SF7, operating without interference; the second scenario

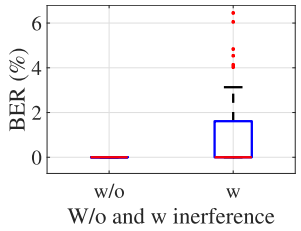


Fig. 3. The BER w/ and w/o interference from other nodes.

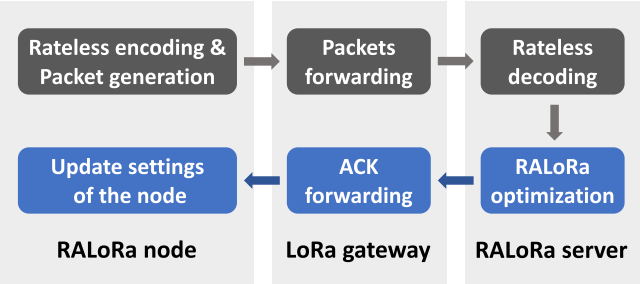


Fig. 4. The architecture of the *RALoRa*.

introduces an interfering node transmitting packets randomly on the same channel and SF. The BER for the underlying node is calculated at six-minute intervals. This node consistently sends packets using channel CH1/SF7. Interference is generated through another node positioned near the gateway, which randomly transmits packets on channel CH1/SF7. As depicted in Figure 3, we observe that the median BER of the underlying node escalates from 0 % to 1.6 % due to the presence of the interfering node. This significant impact underscores the necessity for ADR algorithms to consider in-network interference.

IV. Design of *RALoRa*

This section introduces the design of *RALoRa* from problem formulation to heuristic algorithm development.

A. Overview of RALoRa

Figure 4 depicts the architecture of *RALoRa*, which is composed of three hardware modules: node, gateway, and server. The node partitions the sensing data into multiple original blocks, each of a *block size* S_b. It subsequently generates multiple encoded blocks based on these original blocks and packages them into a packet of size S_p. This packet is sent to the gateway via transmission settings, *i.e.*, *frequency channel* CH, *spreading factor* SF, and *transmission power* P_{tx}. The node then awaits an ACK from the gateway within a short time interval (*e.g.*, one second in our implementation).

After detecting the packet, the gateway demodulates and forwards the packet to the server. The server performs rateless decoding to recover the sensing data and runs an optimization algorithm to determine the node's settings (CH, SF, P_{tx}, S_b, S_p). The settings are then sent to the node via an ACK. The node will apply the new settings in the upcoming sensing cycle for packet transmission.

Favorable network performance in the above process necessitates the joint configuration of resource settings (CH, SF, P_{tx}, S_b, S_p) for all nodes within the LoRa network. Towards this end, we formulate an optimization problem aimed at maximizing the total lifetime of all nodes within the LoRa network (Section IV-B). A rateless-aware network model is established, encapsulating the relationship between the lifetime of the nodes and their resource settings (Section IV-C). To solve the optimization problem, a two-stage heuristic algorithm is introduced in Section IV-D.

B. Problem Formulation

We target on environmental monitoring applications for *RALoRa*. It will accumulate sensing data from various sensors until it reaches a specific size (*e.g.*, 32 bytes). For example, farmers utilize multiple sensors (*e.g.*, temperature, humidity, soil moisture, or soil electrical conductivity sensors) to monitor crop health [28]. Each LoRa node is connected to four sensors, with each sensor producing a sample size of two bytes. Consequently, at each sampling instance, eight bytes of data are collected. *RALoRa* aggregates these samples, collecting four samples (*i.e.*, 32 bytes) in one sensing cycle before transmitting them to the gateway. This data is then encoded into a sequence of rateless blocks using LT codes [11], with block sizes varying between 2, 4, 8, or 32 bytes. Considering the bandwidth for a LoRa frequency channel is 125 kHz and the presence of 64 channels within the 902.3-914.9 MHz range [29], [30], each gateway operates on a unique set of eight channels to avoid interference with neighboring gateways [31]. Therefore, a gateway and its associated nodes constitute an independent network. Given this context, our problem formulation considers a single gateway scenario.

Optimization Objective: In environmental monitoring scenarios, the prioritization of energy-efficient data collection is crucial. Additionally, the nodes in a LoRa network may experience in-network interference, particularly when they operate on the same channel and SF combination (CH/SF). Consequently, the objective of this work is to maximize the total lifetime of all nodes within a LoRa network. This goal is mathematically formulated in Equation (1):

$$\begin{aligned} \max \sum_{i=1}^{N} \frac{LT_i(CH, SF, P_{tx}, S_b, S_p)}{LT_{max}^{SF}} \\ \text{s.t.} \quad CH \in [1,2,3,4,5,6,7,8]\,,\ SF \in [7,8,9,10] \\ P_{tx} \in [2,4,6,8,10,12,14]\,,\ S_b \in [2,4,8,S_d] \end{aligned} \tag{1}$$

where N is the number of nodes in a network, LT_i is the lifetime of node i, which is determined by the settings of nodes. Once the first four parameters are set, packet size S_p can be determined by our rateless-aware network model (Section IV-C). The S_d is the size of sensing data in each cycle, *i.e.*, 32 bytes. LT_{max}^{SF} is the ideal lifetime that a node can achieve using one SF. The ideal lifetime of nodes using SF7 is 5.33 years, while the ideal lifetime of nodes using SF10 is only 0.92 years. If maximizing the total lifetime of all nodes, our heuristic algorithm will focus more on the nodes using small SFs, since their lifetime has a higher influence on the total lifetime. To balance network resources allocated to the nodes using different SFs, we normalize the lifetime of each node by LT_{max}^{SF} and maximize the normalized lifetime of all nodes. Although *RALoRa* focuses on environmental

monitoring applications, it can be applied to other applications as well. The optimization goal would need to be customized to suit these new applications.

The lifetime of a node is the duration from the first startup to the exhaustion of its battery. In this paper, the lifetime is calculated by a linear battery model:

$$LT_i = T_{\text{cycle}} \cdot \frac{E_{\text{battery}}}{E_{\text{cycle}}} \tag{2}$$

where T_{cycle} (the duration of one sensing cycle, *e.g.*, 15 minutes) and E_{battery} (the energy contained in a particular battery) are both constant. E_{cycle} is energy consumed by nodes during each sensing cycle. In Equation (2), we assume an battery model that has linear behavior and does not degrade over time or under environmental influences. Such a liner battery model has been widely adopted in the lifetime calculation of LoRa nodes [4], [15]. There are also some other battery models, such as Electrochemical models, Electrical-circuit models, and Stochastic models [32]. We can also use them to calculate the node lifetime in Equation (2). It will not change the design of *RALoRa*. We can still find settings that achieve fair performance for our application, as long as we use the same battery model for all nodes.

We now calculate E_{cycle}, encompassing the energy consumed by the Radio and microcontroller (MCU) in transmitting (TX), receiving (RX), and sleeping (SP) states. The power consumption of the MCU and Radio in these states is detailed in [15]. Hence, to determine E_{cycle}, we need to calculate the TX time T_{tx} and RX time T_{rx}, with the SP time being $T_{sp} = T_{\text{cycle}} - T_{tx} - T_{rx}$.

The TX time T_{tx} depends on the symbol duration $T_{\text{sym}} = \frac{2^{SF}}{BW}$ (where BW is the bandwidth) and the number of symbols in an uplink packet [33]. An uplink packet consists of N_{pre} preambles, two mandatory sync word symbols, 2.25 start frame delimiter (SFD) symbols, and N_{tx} payload symbols:

$$T_{tx} = T_{\text{sym}} \cdot (N_{\text{pre}} + 2 + 2.25 + N_{tx}) \tag{3}$$

where N_{pre} is typically set to eight as per local regulations [20], and N_{tx} is the number of symbols for a payload size S_p [33]:

$$N_{tx} = 8 + \max\left(\left\lceil \frac{8S_p - 4SF + 28 + 16 - 20\text{h}}{4(SF - 2\text{de})} \right\rceil (\text{cr}+4), 0\right) \tag{4}$$

where the "8" is the number of bits in the PHDR (Physical Header), S_p is the payload size in bytes, "16" is the number of bits for the CRC checksum, h denotes the header type, de is set based on T_{sym} duration, and cr is the coding rate, set in accordance with local regulations [20].

The T_{rx} is calculated similarly to T_{tx}, influenced by transmission parameters and payload size S_p, as described in Equations (3) and (4). The RX mode settings comply with local regulations [20], and the packet size is consistent, typically two bytes (refer to Section IV-D for details).

In the TX mode with our rateless-enabled uplink transmission, each packet's payload includes multiple encoded blocks of size S_b. The number of the encoded blocks depends on the number of original blocks S_d/S_b and the transmission settings (CH, SF, P_{tx}). This interrelation is encapsulated in our rateless-aware network model.

C. Rateless-Aware Network Model

In this section, we develop a network model designed to ascertain the payload size (S_p) for specific transmission settings (CH, SF, P_{tx}) and a given coding setting (block size S_b). This model integrates two key components: rateless-enabled link transmission and bit-level network modeling. The bit-level network module is responsible for calculating the BER, taking into account in-network interference that arises from nodes' transmission settings under ALOHA-based MAC protocols. Subsequently, the rateless-enabled link transmission module utilizes the computed BER and the given block size S_b to determine the optimal payload size.

1) Rateless-Enabled Link Transmission: We generate an uplink packet that contains a calculated number of rateless-encoded blocks. The determination of this number hinges on the probability of block losses, quantified by the BLRR. The BLRR is computed from the BER and the block size S_b:

$$BLRR = (1 - BER)^{(S_b+\epsilon)\cdot 8} \tag{5}$$

where we assume BER is known, which is computed in the next subsection. For each block, we add a CRC checksum of ϵ bytes. We use CRC-4/ITU CRC ($\epsilon = 0.5$), which can detect errors in the block with a high probability (98%), if the block size is smaller than 256 bytes [34].

Based on the estimated BLRR, we can calculate the needed number of blocks N_b in a packet for the sensing data:

$$N_b = \left\lceil \frac{\frac{S_d}{S_b} + \eta}{BLRR} \right\rceil \tag{6}$$

where S_d is the size of sensing data, and η denotes the rateless decoding overhead. The numerator indicates the number of encoded blocks required to successfully decode the data at the server. We set η to 1, *i.e.*, the server needs one extra encoded block for successful decoding.

In rateless-enabled link transmissions, payload size S_p is calculated by multiplying the number of blocks N_b by the sum of block size S_b and the CRC overhead of each block ϵ:

$$S_p = \lceil (S_b + \epsilon) \cdot N_b \rceil \tag{7}$$

If N_b is an odd number, $\epsilon \cdot N_b$ will result in half a byte. We will use one byte for this case with four useless bits.

Disabling Rateless Coding: Upon traversing all block size options, if the computed total lifetime is shorter without utilizing rateless coding, the payload size is set as the sensing data size, *i.e.*, $S_p = S_d$, thereby disabling rateless coding.

Efficient Encoding Coefficient Matrix: Decoding packets using the Gaussian Elimination (GE) algorithm [10] requires the binary encoding coefficient matrix I, which transmitters generate using a random number generator with a predetermined seed. This matrix dictates the formation of rateless blocks, where the original blocks, corresponding to '1' in a matrix row, are XORed to create an encoded block. For example, a row $[1, 0, 1, 0]$ in a 4-block matrix signifies that the first and third blocks are XORed to produce an encoded block. To minimize transmission overhead, *RALoRa* avoids sending the seed with the data packet. Instead, a fixed seed is employed by both senders and receivers to reproduce the matrix I. This is complemented by storing a predefined 32-row matrix I in RAM, enhancing decoding speed and operational efficiency.

2) Bit-Level Network Modeling: In this subsection, we develop the bit-level network modeling to compute the BER for rateless-enabled LoRa link transmission in LoRa networks, which is the only parameter presumed to be known in the rateless-enabled link transmission module.

LoRa links are exposed to two sources of interference, *i.e.*, ambient noise and the in-network interference from other LoRa nodes that are transmitting concurrently using the same channel and SF (CH/SF). The quality of LoRa links can be quantified by Signal-to-Interference-and-Noise Ratio (SINR). With SINR is known, we can calculate BER [35]:

$$BER \approx 0.5 \cdot Q\left(\frac{\sqrt{\mathrm{SINR}\cdot(\mathrm{M}+1)}-\left(\left(H_M\right)^2-\frac{\pi^2}{12}\right)^{\frac{1}{4}}}{\sqrt{H_M-\sqrt{\left(H_M\right)^2-\frac{\pi^2}{12}}+0.5}}\right) \tag{8}$$

where $M = 2^{SF} - 1$, $H_M = \sum_{k=1}^{M} \frac{1}{k}$ denotes the M-th harmonic number. $Q(x) = \frac{1}{\sqrt{2\pi}} \cdot \int_x^{\infty} \exp\left(-\frac{y^2}{2}\right) dy$ is the Q-function. Considering that LoRa is the orthogonal signaling, 0.5 before the Q-function means that for a symbol error, only half of the bits in the symbol could be in error [35]. The Q-function part models the probability that the magnitude of interference envelopes is larger than the magnitude of LoRa signal of interest envelopes.

To determine the BER, it is essential to compute the SINR. However, the [35] does not specify the methodology for computing SINR. Therefore, in this work, we will calculate the SINR by considering both ambient noise and in-network interference. Additionally, our model uniquely incorporates the variable packet size feature of rateless-enabled transmission.

The SINR is defined as the ratio of the signal power received by the gateway from the underlying node i to the in-network interference power and noise power:

$$SINR = \frac{P_{rx}^i}{P_{intra}^i + P_{noise}^i} \tag{9}$$

where P_{noise}^i represents the power of additive white Gaussian noise with zero-mean [36]. The received signal power P_{rx}^i and the in-network interference power P_{intra}^i are calculated by Equations (10) and (11):

$$\begin{aligned} P_{rx}^i &= P_{tx}^i + G_{tx} + G_{rx} - PL\left(d^i\right) \\ PL\left(d^i\right) &= \overline{PL}\left(d_0\right) + 10 \cdot \beta \cdot \log\left(\frac{d^i}{d_0}\right) + X_\sigma \end{aligned} \tag{10}$$

where P_{tx}^i denotes the transmission power of the node i in dBm. The G_{tx} and G_{rx} represent the transmitting and receiving antenna gains, respectively. The $PL\left(d^i\right)$, given in dB, signifies the path loss for the distance d^i between a LoRa node and a gateway. This path loss, characterized by a Gaussian distribution with a mean value of $\overline{PL}(d_0) + 10 \cdot \beta \cdot \log\left(\frac{d^i}{d_0}\right)$ and a standard deviation of X_σ, follows the Log-Normal Shadowing model [1], [37], [38]. We adopt the mean as the quantified path loss. The parameter β is the path loss exponent (PLE). The reference path loss, $\overline{PL}(d_0)$, is pre-measured at a reference distance of $d_0 = 1\,\mathrm{m}$, recording a value of $79.8\,\mathrm{dB}$ in our testbed. This value aligns with measurements from existing empirical studies [38], [39], [40]. For example, Bor et al. reported a path loss of $127.41\,\mathrm{dB}$ at a distance of $40\,\mathrm{m}$ [39] and $\beta = 2.08$. Utilizing Equation (10), we can derive a reference path loss of $94.1\,\mathrm{dB}$ at $d_0 = 1\,\mathrm{m}$. The necessity for the pre-measurement stems from the complexity of signal propagation within the antenna's near-field region, defined as $\leq \frac{2L^2}{\lambda} = \frac{2\times(0.2)^2}{(3\times10^8)/(903.9\times10^6)} = 0.24\,\mathrm{m}$, where L is the antenna dimension and λ is the wavelength. Measuring a known path loss value at d_0 in the antenna's far-field region establish a basis for modeling how path loss varies with distance beyond the near-field. During field deployment, distances are recorded using a GPS logger in the server, enabling the update of β based on the measured SNR and BER (details in Section IV-D3). The Log-Normal Shadowing model is selected for its effectiveness in fitting β, using the least square algorithm on the collected data [1], [38].

In-Network Interference Power P_{intra}^i: We focus on the scenario where a single interfering node transmits a packet concurrently with node i, using the same CH/SF. This consideration is based on the low probability of two or more interfering nodes transmitting simultaneously with node i, as detailed in Section IV-C3. Assuming there are $N_{c,s}$ nodes using the same CH/SF, one of these is the underlying node i, and among the remaining $N_{c,s}-1$ nodes, one is designated as the interfering node j. The expected in-network interference power, P_{intra}^i, is then computed by considering the collision probability and the average interference power contributed by these $N_{c,s} - 1$ nodes.

$$P_{intra}^i = p_k \cdot \sum_{j=1}^{N_{c,s}-1} \left(\frac{P_{rx}^j \cdot T_{intra}^{i,j}}{T_{tx}^i} \cdot \frac{1}{N_{c,s}-1}\right), \quad k=1 \tag{11}$$

where p_k is the probability that a packet sent by the underlying node i collides with k packets transmitted by interfering nodes. To calculate the average interference power, we sum the power contributions from all potential $N_{c,s}-1$ interfering nodes and then divide this total by $N_{c,s} - 1$. For any single interfering node j, its interference power is determined by dividing its interference energy by the transmission time T_{tx}^i. The interference energy is calculated as $P_{rx}^j \cdot T_{intra}^{i,j}$, where P_{rx}^j is the received signal power from node j, and $T_{intra}^{i,j}$ is the overlap duration between the packet transmissions from node i and node j.

The Duration of the In-network Interference $T_{intra}^{i,j}$: Given that the transmission from an interfering node overlaps with the transmission of the underlying node randomly, we calculate the duration of the intra-interference $T_{intra}^{i,j}$ as the expected number of overlapped symbols multiplied by the symbol duration T_{sym}^i:

$$T_{intra}^{i,j} = T_{sym}^i \cdot \sum_{s=1}^{\min\left(N_p^i, N_p^j\right)} \left(s \cdot \frac{1}{\min\left(N_p^i, N_p^j\right)}\right) \tag{12}$$

where N_p^i and N_p^j represent the number of symbols in the packets from nodes i and j, respectively. The number of overlapped symbols, denoted as s, can vary from 1 to the minimum of N_p^i and N_p^j, expressed as $n = \min(N_p^i, N_p^j)$. To estimate the expected number of overlapped symbols,

we calculate the sum of the products of each possible overlap value and its corresponding probability $1/n$. The duration of intra-packet interference, $T^{i,j}_{intra}$, is then obtained by multiplying this expected number of overlapped symbols by the symbol duration T^i_{sym}. It is important to note that the symbol duration for both nodes i and j is identical since they are operating with the same SF and bandwidth.

Packet Collision Probability for ALOHA Protocol p_k: As the analysis of the unslotted ALOHA protocol [7], we assume the packet arrives as a Poisson process [41] with a cumulative arrival rate $\lambda = N_{c,s}/T_{cycle}$. In each sensing cycle, the probability that the underlying packet collides with k interfering packets can be calculated:

$$p_k = \left[\frac{(\lambda \cdot T_{VUL})^k}{k!}\right] \cdot e^{-(\lambda \cdot T_{VUL})}$$
$$T_{VUL} = T^i_{tx} + \sum_{j=1}^{N_{c,s}-1} \left(T^j_{tx} \cdot \frac{1}{N_{c,s}-1}\right) \quad (13)$$

where T_{VUL} is the vulnerable period. The vulnerable transmission time can be the sum of the transmission times from the underlying node i and the interfering node j. Hence, T_{VUL} is computed as the sum of the transmission time of the underlying packet, T^i_{tx}, and the expected transmission time of an interfering packet. In the scenario with one interfering node among N_{cs-1} potential interferers, the transmission time of the interfering packet is averaged across all N_{cs-1} nodes. Consequently, during T_{VUL}, the probability of an interfering node transmitting a packet obeys a Poisson distribution, parameterized by $(\lambda \cdot T_{VUL})$.

3) Discussion: In this section, we discuss some design details of our rateless-aware network model.

Multiple Interfering Nodes: In Equation (11), we do not account for multiple interfering nodes due to their low probability of occurrence. For instance, in a LoRa network with 1000 nodes, each node has the option to select from eight channels and four SFs. We examine two scenarios: one where channels and SFs are evenly distributed among the nodes, and another where nodes are evenly distributed across eight channels but use only one SF. The collision probability in both cases, calculated using Equation (13), assumes a 15-minute sensing cycle and a 32-byte packet size. In the first scenario, the likelihood of having one or fewer interfering nodes transmitting simultaneously with a given node is as high as 98.4 %. In the second scenario, this probability remains significant at 94.1 %. Based on these probabilities, our model primarily considers scenarios with only a single interfering node transmitting concurrently with the underlying node.

Goodput: While our primary goal focuses on maximizing network lifetime, crucial for monitoring applications, we acknowledge that some applications may prioritize goodput. It is an essential metric for applications prioritizing efficient data transmission. Goodput for a node is calculated as follows:

$$GD_i = \frac{S_d}{T_{tx} + T_{rx}} \quad (14)$$

where S_d represents the size of the sensing data, and T_{rx} denotes the time spent in the RX state. The denominator represents the total duration from the start of transmitting the sensing data to the end of receiving an ACK.

A comparison between the calculations of network lifetime and goodput reveals a shared characteristic: both are inversely proportional to the transmission time (T_{tx}). This implies that a shorter T_{tx} can concurrently lead to an extended network lifetime and elevated goodput. Consequently, *RALoRa* optimizes both these metrics by minimizing T_{tx}. Detailed experimental results demonstrating this optimization in Section V-C.

Packet Reception Capabilities of Gateways: The hardware of current LoRa gateways is limited to receiving up to eight packets across different CHs/SFs simultaneously [15], [26]. This limitation implies that a gateway cannot process a ninth packet if it is already handling eight simultaneous transmissions. To assess the impact of this constraint, we calculate the probability of more than eight nodes transmitting at the same time in a LoRa network consisting of 1000 nodes and a single gateway. The probability is only 0.062 %. Given this low probability, the hardware limitations of the gateway are not considered in our rateless-aware network model.

Impact of Header Reception under Interference on *RALoRa*: *RALoRa* utilizes commercial LoRa gateways do not integrate the advanced packet detection algorithms. This limitation can lead to the gateways failing to detect packets in the event of preamble collisions, consequently preventing them from issuing an ACK or NAK. In situations where a node does not receive an ACK or NAK, the node follows the ALOHA protocol's retransmission rules as specified in LoRaWAN [7], [20]. Specifically, if there is no ACK or NAK response within the ACK_TIMEOUT period, defined as two seconds plus a random delay of one to three seconds, the node is programmed to retransmit the packet after a certain back-off period.

BER Equation Selection: Our choice to use Equation (8) for BER estimation over the method in [42] stems from balancing accuracy with computational feasibility. The alternative approach involves intense computations, especially for higher SF. For example, at SF10, it requires over 1 million Q function calculations per node. In a 1000-node network, this escalates to more than a billion calculations per iteration, becoming impractical for frequent iterations. Conversely, the chosen method from [35] needs only one Q function calculation per node, greatly reducing computational burden and aligning with the practical necessity of adhering to the one-second ACK timing constraint in real-world scenarios.

D. Two-Stage Heuristic Algorithm

In this section, we first prove that the optimization problem presented in Equation (1) is NP-complete. A two-stage heuristic algorithm is then proposed to solve the problem.

1) Complexity Analysis: For a LoRa network with N nodes, there are $(8 \times 4 \times 7 \times 4)^N$ possible settings for all nodes, where the four numbers represent the number of optional values for CH, SF, P_{tx}, and S_b, respectively. One gateway can support eight channels. Based on local regulations [20], we can use four SFs, *i.e.*, from SF7 to SF10. When CH, SF, and S_b are constants, our optimization solution aims to determine the transmission power (P_{tx}) for each node in the network. This simplified optimization problem can be analogized to the Partition problem. The Partition problem

involves dividing a set of numbers into two groups such that the sums of the numbers in each group are as close to equal as possible. In our context, each number in this analogy represents the transmission power (P_{tx}) of a LoRa node. The objective is to adjust these power levels in a way that balances the total power usage across the network while also fulfilling the SNR requirements for efficient data transmission. Therefore, our optimization task of setting P_{tx} for each node, while maintaining SNR constraints, can be regarded as a variant of the Partition problem. This analogy underscores the inherent complexity of our optimization task, as it inherits the NP-completeness characteristic of the Partition problem.

Algorithm 1 Offline Optimization

Input : All nodes and the gateway deployment
Output: The configuration of all nodes $\mathbb{CF}$

1 $\mathbb{CF}$ = Random( );
2 LT = Lifetime($\mathbb{CF}$);
3 LT_{old} = LT, $\delta = 0.01$;
4 **while** *True* **do**
5 **for** *each node* $i \in N$ **do**
6 **for** *each* $cf = (CH, SF, P_{tx}, S_b)$ **do**
7 $cf_{\text{temp}} = \mathbb{CF}_i$, $\mathbb{CF}_i = cf$;
8 LT_{temp} = Lifetime($\mathbb{CF}$), $\mathbb{CF}_i = cf_{\text{temp}}$;
9 **if** $LT_{temp} > LT$ **then**
10 $LT = LT_{\text{temp}}$, $\mathbb{CF}_i = cf$;
11 **end**
12 **end**
13 **end**
14 di = abs(LT - LT_{old}), LT_{old} = LT;
15 **if** $di \leq \delta$ **then**
16 break
17 **end**
18 **end**

2) Offline Optimization for Network Initialization: We propose Algorithm 1 to initialize the settings of all nodes when the network is first deployed. During each iteration, each node traverses all of its possible settings (lines 5-12). If an alternative settings for a node results in a longer total lifetime than its current setting, we proceed to update the node's settings (lines 7-11). After one iteration, we assess the improvement in total lifetime by comparing the current iteration's total lifetime with the previous iteration's (line 14). If the improvement exceeds a threshold δ, we infer that there is potential to further enhance the total lifetime in the subsequent iteration; otherwise, we halt the iteration (lines 15-17).

3) In-Situ Updating Algorithm: We also develop Algorithm 2, an in-situ updating algorithm, for updating the configuration of nodes to adapt to dynamic link quality. When the server receives a packet from node i during the current sensing cycle, it employs a KF-based PLE predictor to estimate the PLE β for the next sensing cycle (details in the following paragraph). The estimated β influences the SINR and subsequently the BER values in our rateless-aware network model, ultimately affecting the total lifetime. To accommodate the changed PLE (the measure of the LoRa link quality), we traverse all settings of node i, seeking new settings that maximizes the total lifetime.

KF-based PLE Predictor: Our predictor first measures PLE $\hat{\beta}_k$ using the SNR and BER of the received packet in the current sensing cycle k, which will then be utilized to predict PLE $\hat{\beta}^-_{k+1}$ for the next cycle $k+1$.

Algorithm 2 In-Situ Updating

Input : The current configuration of all nodes - $\mathbb{CF}$
Output: The updated configuration of node i - $\mathbb{CF}_i$

1 **if** *The server receives a packet from node* i **then**
2 LT_{cur} = NodeLife($\mathbb{CF}$) via Equation (2);
3 Update the PLE of the link (Section IV-D3);
4 Update the expected SINR (Section IV-C2);
5 **for** *each configuration* $cf = (CH, SF, P_{tx}, S_b)$ **do**
6 $cf_{\text{temp}} = \mathbb{CF}_i$, $\mathbb{CF}_i = cf$;
7 LT_{temp} = NodeLife($\mathbb{CF}$);
8 **if** $LT_{temp} > LT_{cur}$ **then**
9 $\mathbb{CF}_i = cf$;
10 break
11 **end**
12 **end**
13 **end**

First, we compute the Kalman Filter's input, *i.e.*, $\hat{\beta}_k$, based on the measured SNR and BER. After retrieving the sensing data through rateless decoding, we can obtain the ground truth for all received bits by performing the same rateless encoding as the sender. Consequently, the actual BER can be determined by comparing the received bits with their ground truth. We then compute $\hat{\beta}_k$ by inversely solving Equations (9)(10) based on the measured SNR. Knowing the BER of the received packet allows us to derive another value of $\hat{\beta}_k$ by inversely solving Equations (8)-(10). Finally, the measured value of $\hat{\beta}_k$ is determined by averaging these two calculated $\hat{\beta}_k$.

The Kalman Filter iteratively executes correction and prediction stages to estimate PLE $\hat{\beta}^-_{k+1}$ for the next cycle $k+1$. At cycle k, we have predicted PLE $\hat{\beta}^-_k$ obtained from previous cycle's prediction stage, and the measured $\hat{\beta}_k$ acquired from the SNR or BER at cycle k. In the correction state of cycle k, the corrected PLE is then computed by weighted combining the predicted $\hat{\beta}^-_k$ and measured $\hat{\beta}_k$. The corrected PLE serves as the estimated PLE for the next cycle ($\hat{\beta}^-_{k+1}$). In the configuration of the Kalman Filter, specific values are assigned to the process noise (Q) and the measurement noise (R). The Q is set at $1e^{-4}$, reflecting the anticipated fluctuation level of the PLE over time. The R, determined as 0.1^2, indicates the level of trust in the accuracy of the PLE measurements received in each cycle. We omit the mathematical equations and refer readers to [14] for more details on the Kalman Filter.

To estimate the initial PLE for a specific location, we utilize the Received Signal Strength Indicator (RSSI) values, denoted as P^i_{rx} in Equation (10). These values are obtained from several packets received by the gateway from the node. We then apply the least squares algorithm to the collected RSSI values for fitting the PLE.

Figure 5 illustrates the relative error in BER estimation using four predictive methods. The relative error is defined as

$|\text{Predicted BER} - \text{Groundtruth BER}| / (\text{Ground Truth BER} + \text{offset})$, where the offset is set to $1e^{-4}$. This offset addresses cases when the ground truth BER is zero, which corresponds to a packet delivery rate of 99.68 % [6]. The baseline methods evaluated include Heuristic BER (HB), Heuristic SNR (HS), and Heuristic Path Loss Exponent (HP). These methods estimate BER, SNR, and PLE using a weighted combination of their three most recent values, similar to the methodology described in Figure 2. Packets are chronologically sorted based on their reception time at the gateway, and a sliding window of size one is used to group every four packets for training and testing. The datasets are then split into training and testing sets in a 7:3 ratio. The training set is utilized to optimize the weight values for these methods through a brute-force search. These optimized values are subsequently employed to predict BER, SNR, and PLE for the testing set. Additionally, Figure 5 demonstrates the application of our KF-based PLE predictor. This method uses the measured SNR and BER of the current packet to predict the PLE of the subsequent packet. The estimated PLE is then utilized to calculate the predicted BER. The relative error for each received packet is calculated based on these heuristic methods. Our KF-based PLE predictor outperforms HB, HS, and HP by 47.6 %, 33.4 %, and 27.9 % respectively. This performance is attributed to the KF method's ability to account for current data and its effectiveness in handling interference from other LoRa nodes.

Meeting the ACK Timing Constraint: While ideally we would need to update the settings of all nodes upon receiving packets from any node i, this process is time-consuming and impractical. Thus, our updating algorithm only traverses the available settings of node i, rather than all nodes. We conduct experiments with varying numbers of nodes, Even with 800 nodes, the average execution time is only 796 ms, which is less than the ACK constraint, *i.e.*, a node needs to reply to an ACK within one second (details in Section V-F).

Minimizing Downlink Traffic: *RALoRa* efficiently conserves downlink channel resources by incorporating update information within ACK packets sent to sensor nodes. This strategy is supported by several key factors: 1) LoRaWAN supports both confirmed-data messages, which require ACKs, and unconfirmed-data messages [7], [43], [44]. Confirmed-data messages are vital in scenarios demanding high communication reliability, as they ensure data integrity with CRC checks [5], [44]. For Class A nodes, receiving an ACK within the first or second receiving window is mandatory before sending the next packet, making ACK transmissions a standard procedure rather than an additional requirement unique to our system. 2) In the U.S., LoRaWAN does not impose restrictions on downlink traffic [7], [20]. There are eight available channels for downlinks, ranging from SF7 to SF12, each using a 500 kHz bandwidth [31], which facilitates efficient downlink communication. 3) LoRa nodes are capable of reliably receiving ACK packets [20]. If a node misses an ACK in the first receiving window, it has a second window, utilizing SF12 at the lowest data rate, to ensure reception.

We also minimize the data size needed for updating settings to conserve downlink resources. First, the update information is encoded within two bytes, significantly smaller than the uplink packet size. This compact encoding involves allocating

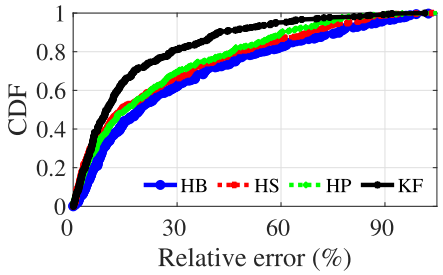


Fig. 5. The CDF of the BER prediction error for four predictive methods.

three bits for CH, two bits for SF, three bits for P_{tx}, two bits for S_b, and six bits for N_b. The parameter S_p can be inferred from S_b and N_b. Second, when a node's settings do not require modification, *RALoRa* leaves ACK packets unchanged. Field experiments show that, on average, only 43.6 % of sensing cycles require an update to the nodes' settings.

4) Retransmission Mechanism: RALoRa optimizes packet size to achieve successful rateless decoding with a single transmission from LoRa nodes. Accurate BER estimation is essential for determining the optimal packet size. Our link quality predictor enhances BER prediction accuracy by 47.6 %, but a potential error margin of up to 30.9 % remains, which could result in suboptimal packet sizing. In cases where the server cannot recover sensing data from the first transmission, *RALoRa* employs a HARQ mechanism. This mechanism facilitates the transmission of additional encoded blocks, which are combined with correctly received blocks from the first transmission for rateless decoding.

Upon receiving a NAK specifying the number of additional blocks required, the sender retransmits after a randomized delay, following LoRaWAN guidelines [7], [43]. This delay mitigates collision risks by diversifying retransmission times among nodes. Each node's retransmission schedule is determined by a pseudo-random generator, seeded with the node's unique address. This strategy of retransmitting only the necessary additional blocks, rather than the entire packet, substantially improves transmission efficiency. It also offers more effective management of duty cycle constraints compared to conventional LoRaWAN protocols, reducing the volume of data transmitted during retransmissions.

V. Evaluation

A. Experimental Testbed

The LoRa nodes are custom-built using the SX1276 Radio [45] on the Arduino Uno board [46]. Each node is powered by a 3,000 mAh power bank and operates within the 903.9-905.3 MHz frequency band, utilizing a bandwidth of 125 kHz. Both nodes and gateways employ omni-directional antennas, with gains of 5 dBi for nodes and 3 dBi for gateways, respectively. The transmission power of the LoRa nodes can vary from 2 to 14 dBm in 2 dBm increments. We integrate LT encoding into the LMIC library [47]. The gateway runs a LoRa Packet Forwarder thread [48] that demodulates packets for server transmission or relays ACK/NAK messages to nodes. In the gateway's configuration file, we set the `forward_crc_error` field to `true` for forwarding corrupted packets to the server.

We implement the *RALoRa* server utilizing an open-source library tailored for LoRaWAN [49]. The server is operational on a Lenovo ThinkPad X1 Carbon laptop, equipped with an Intel (R) Core (TM) i7-4600 CPU at 2.69 GHz. Our algorithms are implemented using Python scripts, which interface with the server via WebSocket for seamless communication. Both uplink and downlink packets are logged in a MongoDB database. This database enables our algorithms to efficiently access and modify the settings of all nodes in the network, facilitating dynamic network management and optimization.

In our in-field experiments, a LoRa network comprising ten nodes and one gateway is deployed in a residential area in the U.S., depicted in Figure 6. During the experiments, we observed negligible external interference from other users in the 900 MHz band affecting our testbed. To emulate a larger network of 450 nodes, we employed a strategy of reducing the sensing cycle duration from the standard 15 minutes to 20 seconds [15]. As a result, each node in our setup transmits 45 packets every 15 minutes, replicating the transmission frequency of a 450-node network. For the implementation of the 15-minute sensing cycle, the first received packet from each node is utilized to predict the BER of the 46th packet, as part of our in-situ updating algorithm. This process is repeated sequentially. Thus, our experimental setup effectively simulates a LoRa network comprising 450 nodes operating on a 15-minute sensing cycle.

The average distance between the nodes and the gateway is 308.2 meters, with the maximum and minimum distances being 516.6 meters and 185.9 meters, respectively. The reasons for the limited communication range are fourfold. **First**, the testbed's deployment in a residential area, characterized by diverse landcover types such as trees, buildings, roads, and parked cars, poses significant obstacles. These obstructions lead to considerable signal attenuation, adversely impacting the SNR of LoRa packets. Similar observations about LoRa's reduced communication range in complex environments have been reported [4], [25], [38]. **Second**, the height of the antennas plays a crucial role in signal reception [50]. In our setup, the LoRa nodes are positioned on the ground, while the gateway is situated adjacent to a second-floor window, about 3 meters above ground level. These two factors collectively result in a high propagation path loss. **Third**, compliance with local regulations restricts us to using only SF between SF7 and SF10 [20]. Although employing SF12 could extend the communication range, its usage is not permissible in our testbed area. **Fourth**, the SNR range of received packets in our testbed spans extensively from -20 dB to 5 dB, as illustrated in Figure 7(a). This broad range results from varying distances of the nodes from the gateway, coupled with physical obstructions like trees, buildings, or vehicles. Encompassing the lowest SNR limit for SF10, this SNR spectrum provides a thorough basis for evaluating *RALoRa* under diverse SNR conditions.

B. Experimental Setup

We uses NS-3 simulations and in-field experiments to validate *RALoRa*. The in-field experiments offer real-world system validation, while NS-3 simulations focus on individual component assessment within controlled settings. In-field experiments track link quality variations over time, testing the entire system performance. Conversely, NS-3 simulations involve a smaller 10-node network for examining the Rateless-Enabled Link Transmission in low-interference conditions, and a larger 800-node network to assess the Bit-Level Network Modeling under high-collision scenarios. The experiments progress in three phases: first, assessing overall system performance in the field (Section V-C); second, evaluating component efficacy and performance in varied large-scale network settings (Sections V-D and V-E); and finally, analyzing system overhead (Section V-F). The sensing data size remains constant at 32 bytes across all experiments.

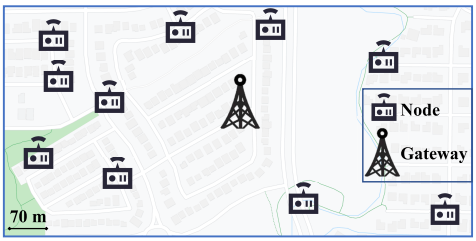


Fig. 6. The experimental testbed layout in a residential area.

1) Benchmarks: We compare the performance of *RALoRa* with the following two protocols.

LoRaWAN [7] is the standard protocol used in LoRa networks. The default ADR mechanism is enabled. It chooses the smallest available SF based on the predicted SNR.

eLoRa [13] also incorporates block-level LT codes into LoRa networks. It employs a LoRa link model to determine the block size and SF of a LoRa node, relying on SNR measurements of recently received packets.

2) Performance Metrics: This paper focuses on the environmental monitoring applications that need nodes to provide high energy efficiency (lifetime) and reliability (data yield). A long lifetime reduces battery replacement, and a high data yield avoids the loss of sensing data. Meanwhile, we check whether a long lifetime can provide high goodput in in-field experiments. In summary, we use the following three metrics:

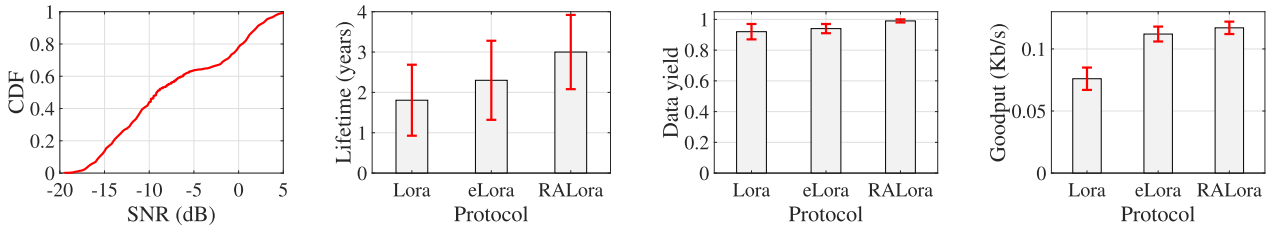
- *Lifetime.* It is the time that the node exhausts its battery, which is calculated by Equation (2). We ignore the energy consumption of rateless encoding as it consumes a little energy, which is measured in Section V-F.
- *Data Yield [10].* The data yield of a node is the ratio between the amount of the sensing data successfully received by the server and the total amount of the sensing data sent by that node.
- *Goodput.* The goodput of a node is the number of sensing data in bits that the LoRa link transmits to the server per second. It is calculated by Equation (14).

We also study these performance metrics at the network level, which is the average of all nodes, *e.g.*, the data yield of a network is the average data yield of all nodes in the network.

C. Overall Performance

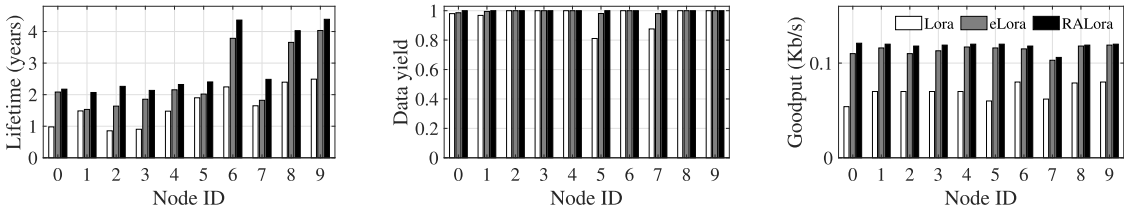
We run each of the benchmark approaches sequentially on the deployed LoRa network. Every experiment lasts for a duration of three hours. The results are presented both at the network and node levels.

1) Network Level Performance: Figure 7 demonstrates the network performance with a standard deviation of LoRaWAN, eLoRa, and *RALoRa*. On average, *RALoRa* achieves the



(a) CDF of SNRs from all locations (b) Lifetime (c) Data yield (d) Goodput

Fig. 7. The overall network performance comparison on lifetime, data yield and goodput in the in-field experiments. Due to space constraints on the x-axis ticks, abbreviations are used in this and some subsequent figures: Lora for LoRaWAN, eLora for eLoRa, and RALora for *RALoRa*.



(a) Lifetime (b) Data yield (c) Goodput

Fig. 8. The performance comparison on the lifetime, data yield, and goodput for each LoRa node in the in-field experiments.

performance improvement over other two protocols on the lifetime, data yield, and goodput. In particular, *RALoRa* increases lifetime of LoRaWAN and eLoRa by 66.1 % and 30.4 %, respectively. It improves the data yield of LoRaWAN and eLoRa by 7.6 % and 5.3 %. It also increases the goodput of LoRaWAN and eLoRa by 53.8 % and 4.5 %. Such performance gain is mainly derived from the rateless-aware network model and KF-based PLE predictor.

Lifetime: Figure 7(b) depicts the network lifetime. Compared with LoRaWAN, eLoRa provides a higher lifetime. This is because eLoRa tends to use a smaller SF than LoRaWAN by adopting rateless coding. Smaller SF provides shorter packet transmission time. In addition, eLoRa does not need to retransmit the entire packet as LoRaWAN does. It just needs to retransmit some extra encoded blocks.

RALoRa provides the highest lifetime. First, the packets for *RALoRa* contain the appropriate number of encoded blocks to compensate for the block transmission errors. The server has a higher probability of successfully recovering sensing data based on the first packet transmission. However, eLoRa uses invariant block numbers in packets, which need to retransmit extra encoded blocks. Second, *RALoRa* considers in-network interference to more reasonably allocate SFs, which can avoid high packet collision. Third, we consider the overhead of rateless coding in rateless-enabled transmissions. *RALoRa* only enables rateless coding if the performance gain provided by rateless-enabled transmission is larger than its overhead. However, eLoRa always enables rateless coding. Finally, our in-situ updating algorithm adapts to changing link quality more accurately than eLoRa via a KF-based PLE predictor.

Data Yield: If the retransmission number is greater than four, nodes will stop transmitting the current sensing data, which causes sensing data losses [7]. Figure 7(c) shows the network data yield. Compared with LoRaWAN, eLoRa exhibits a higher data yield. For eLoRa, each transmission can bring new data information to help to recover sensing data by rateless decoding. Hence, eLoRa can utilize the blocks without error bits from previous transmissions, and next time only transmit extra needed blocks, not the entire packet. However, LoRaWAN abandons the whole packet, then retransmits the packet again. *RALoRa* achieves the highest data yield. This is because the first packet in *RALoRa* already contains enough encoded blocks to compensate for the block losses, which further reduces the number of packet retransmission.

Goodput: Figure 7(d) depicts the network goodput of three protocols. It verifies that goodput exhibits similar trends with lifetime. *RALoRa* still achieves the highest goodput than LoRaWAN and eLoRa. But the performance gain of *RALoRa* is marginal since the ACK time is included when calculating the goodput. The transmission time saved by the *RALoRa* is relatively small compared with the ACK time in the denominator of Equation (14).

2) Node Level Performance: Figure 8 depicts the performance of each node in the LoRa network. We can find that *RALoRa* can improve the energy efficiency and reliability of all the nodes regardless of their location in the network.

Lifetime: As depicted in Figure 8(a), the lifetime of some nodes in LoRaWAN, such as nodes #2 and #3, is notably short. This is because they must employ higher SFs to ensure reliable communication. In contrast, eLoRa tends to use smaller SFs because of the rateless coding. *RALoRa* further enhances the lifetime by proactively adapting to link quality.

Data Yield: Figure 8(b) presents the data yield for all nodes. For certain nodes in LoRaWAN, such as node #5, the data yield is low due to poor link quality. *RALoRa* improves the data yield by reducing retransmissions, achieved by encapsulating the encoded rateless blocks in packets. Additionally, *RALoRa* introduces a network model to allocate SF, subsequently minimizing the probability of packet collisions.

Goodput: Figure 8(c) illustrates the goodput of each node under different protocols. The LoRa nodes that are distant from the LoRa gateway, such as nodes #2, #3, and #5, exhibit the reduced goodput due to their adoptions of larger SFs. Notably, there is a discernible correlation between the goodput and the lifetime: a longer lifetime generally correlates with higher goodput for each node.

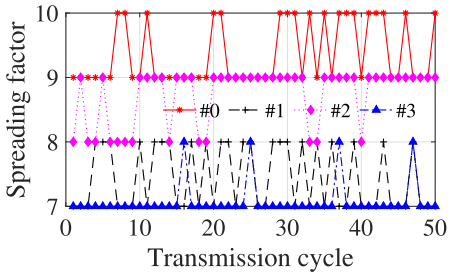


Fig. 9. The adopted SFs for four LoRa nodes over time.

3) SF Updating: Given the dynamic link quality, LoRa nodes need to update their transmission parameters to adapt to the variations. Figure 9 depicts the SFs used by four nodes during in-field experiments. While nodes #0, #1, and #2 frequently adjust their SFs in response to the changing link quality, node #3 consistently employs SF7. This is due to its proximity to the gateway and the fact that SF7 carries the lowest collision risk with other nodes. As a result, LoRa node #3 does not need to change its SF.

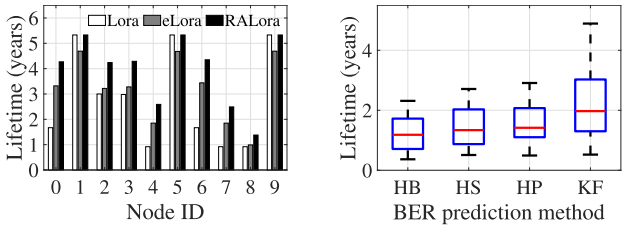
D. The Effectiveness of RALoRa Components

We build two LoRa networks in the NS-3 to verify the effectiveness of our proposed components in *RALoRa*. We utilize an open-source GitHub library [51] to simulate the LoRaWAN on the NS-3 [52]. The modeling of LoRaWAN in NS-3 comprises several components, *e.g.*, the physical layer, and the MAC layer. The details of the implementation can be found in [36]. Based on the library, we implement *RALoRa* and eLoRa in NS-3. All nodes and gateways are located in an area with a radius of 3.3 kilometer. Each node sends 32-byte data at each sensing cycle (15 minutes). Nodes are uniformly distributed in the area. All nodes and gateway are configured to use the same 125 kHz LoRaWAN channel (915 MHz). The PLE of LoRa links is fixed at 3.0. We do not present the data yield and goodput result since similar trends between them have been demonstrated in Section V-C.

1) Rateless-Enabled Link Transmission: To investigate the effectiveness of rateless-enabled link transmission, a LoRa network consisting of 10 nodes and a gateway is built. In this network, the packet collision probability is low, *i.e.*, 0.04 % (Equations (13)). We can assume that there is no in-network interference. Therefore, the performance gain comes solely from the rateless-enabled link transmission.

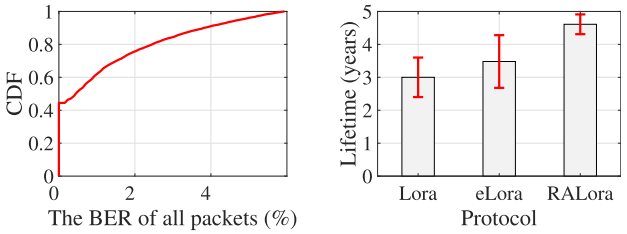
In Figure 10(a), eLoRa provides a longer lifetime than LoRaWAN for most nodes, *e.g.*, eLoRa increases the lifetime of node #2 by 7.3 %. Because eLoRa selects a smaller SF by using rateless coding to compensate for error bits. However, the performance of eLoRa is worse than LoRaWAN for nodes #1, #5, and #9. This is because there are only ten nodes, and the packet collision probability is low. The nodes #1, #5, and #9 use SF7 for their good link quality. The rateless coding in eLoRa does not provide performance gain but adds extra overhead from rateless coding.

RALoRa performs better than eLoRa for all nodes, *e.g.*, 3.44 years *v.s.* 4.35 years for node #6. Although they both use rateless coding, eLoRa has a constant packet size and will retransmit extra blocks to compensate for error blocks. *RALoRa* has fewer retransmissions, for it includes more



(a) Rateless-enabled transmission (b) KF-based PLE predictor

Fig. 10. The effectiveness of the proposed two components.



(a) Link quality (b) Lifetime

Fig. 11. The performance gain of bit-level network modeling.

encoded blocks in the first packet. In addition, for nodes #1, #5, and #9 where eLoRa under-performs LoRaWAN, *RALoRa* achieves comparable performance to LoRaWAN. This is because if rateless coding has no gain, *RALoRa* avoids the overhead of rateless encoding by disabling rateless encoding.

2) KF-Based Link Quality Predictor: Figure 10(b) presents the lifetime of all links using various prediction methods. The KF-based PLE predictor outperforms Heuristic BER (HB), Heuristic SNR (HS), and Heuristic path loss exponent (HP) by 66.8%, 47.3%, and 39.2%, respectively. This enhanced performance is attributed to more accurate link quality prediction, as evidenced in Figure 5. Thus, while the incremental design of our rateless coding significantly boosts efficiency, the crucial role of precise link estimation in optimizing the initial packet transmission is also evident.

3) Bit-Level Network Modeling: In this section, we set up a LoRa network composed of 800 nodes and a single gateway. This network experiences a packet collision probability of 10.6 %, pointing to significant in-network interference. In Figure 11(a), we can observe that 54.7 % of the packets in the network contain bit errors.

The comparative performance of three protocols is illustrated in Figure 11(b). Notably, eLoRa outperforms LoRaWAN in terms of network lifetime. Figure 12 presents the distribution of SFs utilized across the network. It is evident that eLoRa often selects smaller SFs compared to LoRaWAN. For instance, the ratios of SF7 in eLoRa and LoRaWAN are 34.2 % and 29.1 % respectively. The inclination towards smaller SFs consequently reduce packet collision probabilities. This assertion is supported by Figure 13, where eLoRa diminishes collision probability by 33.7 % for nodes operating on SF10, compared with LoRaWAN. Another difference lies in the retransmission strategy: LoRaWAN retransmits the entire packet, whereas eLoRa only retransmits blocks. This selective retransmission by eLoRa requires less transmission time.

The *RALoRa* further boosts the lifetime of eLoRa by 32.8 % by strategically setting SFs to minimize node interference. In *RALoRa*, we noted that the usage ratios for SF7 to SF10 are

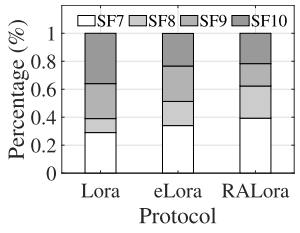


Fig. 12. The used SF distribution for three protocols in a LoRa network.

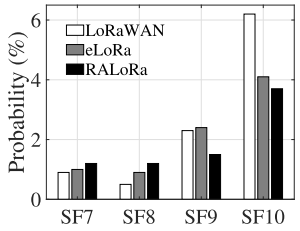


Fig. 13. The packet collision probability at different SFs.

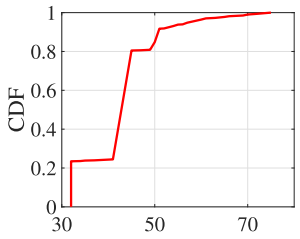


Fig. 14. The adopted packet size in the *RALoRa* protocol.

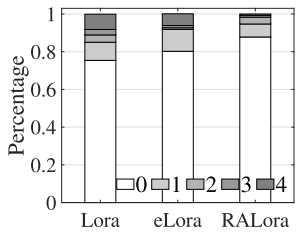


Fig. 15. The number of retransmissions for three protocols.

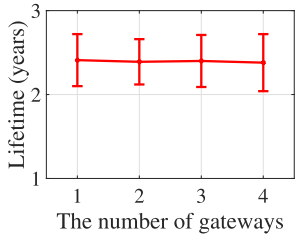


Fig. 16. The impact of the number of gateways on lifetime.

39.3 %, 23.0 %, 16.0 %, and 21.7 %, respectively, as illustrated in Figure 12. The packet sizes within *RALoRa* vary between 32 and 77 bytes, a trend detailed in Figure 14. This flexible packet sizing allows for the inclusion of additional encoded blocks, compensating for potential block losses. Furthermore, as confirmed by Figure 15, *RALoRa* achieves the fewest number of retransmissions in the network.

E. Performance Under Different Settings

We further investigate the performance of *RALoRa* under different experimental settings.

Multiple Gateways: We conduct experiments with multiple gateways to investigate the impact of the number of gateways on the normalized lifetime. Each gateway covers

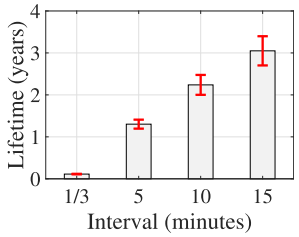


Fig. 17. The impact of the duration of the sensing cycle.

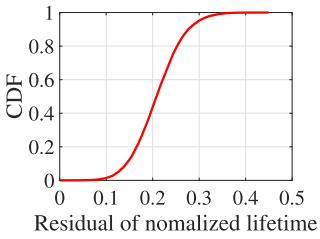


Fig. 18. The residual of the normalized node lifetime.

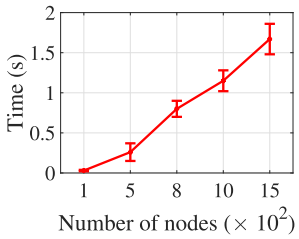


Fig. 19. The execution time for different numbers of nodes.

800 nodes and uses a unique set of eight channels. All of them are placed in an area with a radius of 6.0 kilometer. Figure 16 confirms that the number of gateways almost does not affect the performance of *RALoRa*. Because each gateway and its covered nodes form an independent LoRa network, it would not interfere with other LoRa gateways [31].

Sensing Intervals: In in-field experiments, each node sends 45 packets every 15 minutes with an interval of 20 seconds. To calculate the BER under the 15-minute sensing interval, we use the first packet to predict the BER of the 46th packet, and so on. In this way, we can calculate the BERs at different sensing intervals. Based on the computed BER, the expected lifetime is calculated. Figure 17 displays how the lifetime change as the sensing interval increases. We can find that as the sensing interval increases, the lifetime increases significantly.

F. The Efficiency of RALoRa

We evaluate the optimality of the proposed heuristic algorithms and quantify the energy overhead of *RALoRa*.

1) The Efficiency of Heuristic Algorithms: We present a comparison of the outcomes from our heuristic algorithms against the optimal solution, followed by an assessment of the execution time of algorithms.

The Optimality of Algorithms: To evaluate the optimality of our heuristic algorithm, extensive NS-3 simulations are conducted in a LoRa network setup comprising 20 nodes and a gateway. The nodes are randomly positioned using a different random seed, facilitating a comparison between the heuristic algorithm's solutions and those obtained through brute-force searching. For each simulation, the heuristic algorithm (Algorithm 1) is executed to determine each node's transmission configurations, and the outcomes are compared

with the brute-force optimal solutions. This process is repeated 10,000 times with varying seeds to amass a comprehensive dataset of residual normalized lifetime values. The residual normalized lifetime, defined as the difference in normalized lifetime between the brute-force optimal solution and the heuristic solution, is calculated using Equation (1). Here, lifetime values are normalized against the maximum achievable lifetime for a node using a single SF, with higher values indicating longer network lifetimes. Figure 18 presents the distribution of these values, with a recorded Kurtosis of only 0.13. The Kurtosis is a statistical measure indicating the shape of a distribution [53], [54], [55]. A kurtosis value close to 0, as observed in our results, denotes a distribution shape akin to a normal distribution, suggesting that the performance of our algorithm closely approximates the optimal solutions across diverse network configurations. The mean value of 0.21 further attests to the efficacy of our heuristic algorithm.

Execution Time: In our setup, as the gateway and server are grid-powered, their energy consumption is not a concern. We focused on measuring the execution time of the in-situ updating algorithm across varying node counts. As depicted in Figure 19, execution time increases with the number of nodes. For 800 nodes, the average time is 796 ms, while the receiver's rateless decoding takes an additional 11 ms, totaling 807 ms for both decoding and updating. This duration leaves only 200 ms for ACK transmission, risking insufficiency under heavy internet traffic (ACK threshold is one second). Reducing node number to around 500 cuts the server processing time to about 400 ms, allowing approximately 600 ms for ACK transmission within the first window. For networks still facing timing challenges due to internet delays, two strategies are proposed: 1) Adding more gateways can distribute the load, reducing server processing time and ensuring timely first-window ACK transmission. 2) If the first window proves unfeasible, configuring nodes to use the second ACK window (with a 2-second limit) could be a solution. Although this may affect node lifetime, it ensures reliable ACK reception in networks with notable internet delays.

2) Energy Overhead on Nodes: The energy consumption of a node originates from the following two components:

Generating Encoded Rateless Blocks: We measure the energy consumed in LoRa nodes during the rateless encoding process, which involves encoding sensing data into 32 encoded blocks, each with a size of 8 bytes. During the process, the MCU is in an active state with a power of 23.48 mW [15]. This process takes an average time of 31.2 ms, resulting in an energy consumption of 0.73 mJ. This energy consumption is only $4.1e^{-7}$ % of the total energy of nodes.

Receiving Updated Settings: The LoRa nodes need to receive two bytes of data for updating settings. In a worst case, these two bytes are received in the second window, leading to a two seconds waiting time for the MCU and the data is transmitted using SF12. The reception time is 16.4 ms. The energy consumed is 1.21 mJ, which represents a $6.7e^{-7}$ % of the total battery capacity.

VI. Conclusion

This paper presents a rateless-enabled data rate adaptation scheme for LoRa networks. We formulate an optimization problem to maximize the total lifetime of all nodes via allocation of network resources. A rateless-aware network model is then proposed to compute lifetime by considering rateless-enabled link transmission and in-network interference. Finally, we design a heuristic algorithm to solve the problem. Extensive experiments show the effectiveness of *RALoRa*.

Acknowledgment

Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

References

[1] K. Yang, Y. Chen, X. Chen, and W. Du, "Link quality modeling for LoRa networks in orchards," in *Proc. 22nd ACM/IEEE Conf. Inf. Process. Sensor Netw. (IPSN)*, May 2023, pp. 27–39.

[2] O. Iova et al., "LoRa from the city to the mountains: Exploration of hardware and environmental factors," in *Proc. ACM Int. Conf. Embedded Wireless Syst. Netw. (EWSN)*, 2017, pp. 317–322.

[3] S. Demetri, G. P. Picco, and L. Bruzzone, "LaPS: LiDAR-assisted placement of wireless sensor networks in forests," *ACM Trans. Sensor Netw.*, vol. 15, no. 2, pp. 1–40, 2019.

[4] A. Gadre, R. Narayanan, A. Luong, A. Rowe, B. Iannucci, and S. Kumar, "Frequency configuration for low-power wide-area networks in a heartbeat," in *Proc. 17th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2020, pp. 339–352.

[5] A. Balanuta, N. Pereira, S. Kumar, and A. Rowe, "A cloud-optimized link layer for low-power wide-area networks," in *Proc. 18th Int. Conf. Mobile Syst., Appl., Services (MobiSys)*, 2020, pp. 247–259.

[6] K. Yang and W. Du, "LLDPC: A low-density parity-check coding scheme for LoRa networks," in *Proc. 20th ACM Conf. Embedded Netw. Sensor Syst. (SenSys)*, 2022, pp. 193–206.

[7] L. Alliance. (2017). *LoRaWAN 1.1 Specification*. [Online]. Available: https://lora-alliance.org/wp-content/uploads/2020/11/lorawantm_specification_-v1.1.pdf

[8] J. Álamos, P. Kietzmann, T. C. Schmidt, and M. Wählisch, "Poster: DSME-LoRa—A flexible MAC for LoRa," in *Proc. IEEE 29th Int. Conf. Netw. Protocols (ICNP)*, Nov. 2021, pp. 1–2.

[9] A. Gudipati and S. Katti, "Strider: Automatic rate adaptation and collision handling," in *Proc. ACM Special Interest Group Data Commun. (SIGCOMM)*, 2011, pp. 158–169.

[10] W. Du, Z. Li, J. C. Liando, and M. Li, "From rateless to distanceless: Enabling sparse sensor network deployment in large areas," *IEEE/ACM Trans. Netw.*, vol. 24, no. 4, pp. 2498–2511, Aug. 2016.

[11] M. Luby, "LT codes," in *Proc. 43rd IEEE Annu. Symp. Found. Comput. Sci. (FOCS)*, Nov. 2002, p. 271.

[12] W. Du, J. C. Liando, H. Zhang, and M. Li, "Pando: Fountain-enabled fast data dissemination with constructive interference," *IEEE/ACM Trans. Netw.*, vol. 25, no. 2, pp. 820–833, Apr. 2017.

[13] G. Chen, J. Lv, and W. Dong, "Exploiting rateless codes and cross-layer optimization for low-power wide-area networks," in *Proc. IEEE/ACM 28th Int. Symp. Quality Service (IWQoS)*, Jun. 2020, pp. 1–9.

[14] G. Welch and G. Bishop. (1995). *An Introduction to the Kalman Filter*. [Online]. Available: http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf

[15] J. C. Liando, A. Gamage, A. W. Tengourtius, and M. Li, "Known and unknown facts of LoRa: Experiences from a large-scale measurement study," *ACM Trans. Sensor Netw.*, vol. 15, no. 2, pp. 1–35, May 2019.

[16] W. Gao, W. Du, Z. Zhao, G. Min, and M. Singhal, "Towards energy-fairness in LoRa networks," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2019, pp. 788–798.

[17] S. Fahmida, V. P. Modekurthy, M. Rahman, A. Saifullah, and M. Brocanelli, "Long-lived LoRa: Prolonging the lifetime of a LoRa network," in *Proc. IEEE 28th Int. Conf. Netw. Protocols (ICNP)*, Oct. 2020, pp. 1–12.

[18] W. Gao, Z. Zhao, and G. Min, "AdapLoRa: Resource adaptation for maximizing network lifetime in LoRa networks," in *Proc. IEEE 28th Int. Conf. Netw. Protocols (ICNP)*, Oct. 2020, pp. 1–11.

[19] T. Joachim. (2019). *Complete reverse engineering of LoRa PHY*. [Online]. Available: https://www.epfl.ch/labs/tcl/wp-content/uploads/2020/02/Reverse_Eng_Report.pdf

[20] L. Alliance. (2017). *LoRaWAN 1.1 Regional Parameters*. [Online]. Available: https://lora-alliance.org/wp-content/uploads/2020/11/lorawan-regional-parameters-v1.1ra.pdf

[21] Z. Sun et al., "FLoRa: Energy-efficient, reliable, and beamforming-assisted over-the-air firmware update in LoRa networks," in *Proc. 22nd ACM/IEEE Conf. Inf. Process. Sensor Netw. (IPSN)*, May 2023, pp. 14–26.

[22] Y. Yu, L. Mroueh, G. Vivier, and M. Terré, "Packet recovery latency of a rate-less polar code in low power wide area networks," in *Proc. Eur. Conf. Netw. Commun. (EuCNC)*, Jun. 2019, pp. 10–14.

[23] B. Li, D. Tse, K. Chen, and H. Shen, "Capacity-achieving rateless polar codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2016, pp. 46–50.

[24] W. Song, Y. Shen, L. Li, K. Niu, and C. Zhang, "A general construction and encoder implementation of polar codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 7, pp. 1690–1702, Jul. 2020.

[25] R. Eletreby, D. Zhang, S. Kumar, and O. Yağan, "Empowering low-power wide area networks in urban settings," in *Proc. ACM Special Interest Group Data Commun. (SIGCOMM)*, 2017, pp. 309–321.

[26] A. Gamage, J. C. Liando, C. Gu, R. Tan, and M. Li, "LMAC: Efficient carrier-sense multiple access for LoRa," in *Proc. 26th Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, 2020, pp. 1–13.

[27] C. Li, X. Guo, L. Shangguan, Z. Cao, and K. Jamieson, "CurvingLoRa to boost LoRa network capacity via concurrent transmission," in *Proc. 19th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2022, pp. 879–895.

[28] G. Codeluppi, A. Cilfone, L. Davoli, and G. Ferrari, "LoRaFarM: A LoRaWAN-based smart farming modular IoT architecture," *Sensors*, vol. 20, no. 7, p. 2028, 2020.

[29] J. Liu, J. Gao, S. Jha, and W. Hu, "Seirios: Leveraging multiple channels for LoRaWAN indoor and outdoor localization," in *Proc. 27th Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, 2021, pp. 656–669.

[30] K. Yang, Y. Chen, and W. Du, "OrchLoc: In-orchard localization via a single LoRa gateway and generative diffusion model-based fingerprinting," in *Proc. 22nd ACM Int. Conf. Mobile Syst., Appl., Services (MobiSys)*, 2024, pp. 1–14.

[31] J. Liu, W. Xu, S. Jha, and W. Hu, "Nephalai: Towards LPWAN C-RAN with physical layer compression," in *Proc. 26th Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, 2020, pp. 1–12.

[32] M. R. Jongerden and B. R. Haverkort, "Which battery model to use?" *IET Softw.*, vol. 3, no. 6, pp. 445–457, 2009.

[33] Semtech. (2013). *SX1272/3/6/7/8: LoRa Modem Design Guide*. [Online]. Available: https://www.openhacks.com/uploadsproductos/loradesignguide_std.pdf

[34] P. Koopman and T. Chakravarty, "Cyclic redundancy code (CRC) polynomial selection for embedded networks," in *Proc. IEEE Int. Conf. Dependable Syst. Netw. (DSN)*, Jun./Jul. 2004, pp. 145–154.

[35] T. Elshabrawy and J. Robert, "Closed-form approximation of LoRa modulation BER performance," *IEEE Commun. Lett.*, vol. 22, no. 9, pp. 1778–1781, Sep. 2018.

[36] F. Van den Abeele, J. Haxhibeqiri, I. Moerman, and J. Hoebeke, "Scalability analysis of large-scale LoRaWAN networks in NS-3," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 2186–2198, Dec. 2017.

[37] T. S. Rappaport et al., *Wireless Communications: Principles and Practice*, vol. 2. Upper Saddle River, NJ, USA: Prentice-Hall, 1996.

[38] S. Demetri, M. Zúñiga, G. P. Picco, F. Kuipers, L. Bruzzone, and T. Telkamp, "Automated estimation of link quality for LoRa: A remote sensing approach," in *Proc. 18th ACM/IEEE Int. Conf. Inf. Process. Sensor Netw. (IPSN)*, Apr. 2019, pp. 145–156.

[39] M. C. Bor, U. Roedig, T. Voigt, and J. M. Alonso, "Do LoRa low-power wide-area networks scale?" in *Proc. 19th ACM Int. Conf. Modeling, Anal. Simulation Wireless Mobile Syst. (MSWiM)*, 2016, pp. 59–67.

[40] A. Alsayyari, I. Kostanic, and C. E. Otero, "An empirical path loss model for wireless sensor network deployment in a concrete surface environment," in *Proc. IEEE 16th Annu. Wireless Microw. Technol. Conf. (WAMICON)*, Apr. 2015, pp. 1–6.

[41] M. O. Shahid, M. Philipose, K. Chintalapudi, S. Banerjee, and B. Krishnaswamy, "Concurrent interference cancellation: Decoding multi-packet collisions in LoRa," in *Proc. ACM Special Interest Group Data Commun. (SIGCOMM)*, 2021, pp. 503–515.

[42] T. Elshabrawy and J. Robert, "Analysis of BER and coverage performance of LoRa modulation under same spreading factor interference," in *Proc. IEEE 29th Annu. Int. Symp. Pers., Indoor Mobile Radio Commun. (PIMRC)*, Sep. 2018, pp. 1–6.

[43] L. Alliance. (2022). *LoRaWAN for Developer*. [Online]. Available: https://lora-alliance.org/lorawan-for-developers

[44] X. Xia, Q. Chen, N. Hou, Y. Zheng, and M. Li, "XCopy: Boosting weak links for reliable LoRa communication," in *Proc. 29th Annual Int. Conf. Mobile Comput. Netw. (MobiCom)*, 2023, pp. 1–15.

[45] Semtech. (2020). *Semtech SX1276 Datasheet*. [Online]. Available: https://www.semtech.com/products/wireless-RF/lora-transceivers/sx1276

[46] Arduino. (2021). *Arduino Uno Rev3*. [Online]. Available: https://store-usa.arduino.cc/products/arduino-uno-rev3/?selectedStore=us

[47] M. Kooijman and T. Telkamp. (2021). *LMIC Library*. [Online]. Available: https://github.com/mcci-catena/arduino-lmic

[48] Semtech. (2017). *LoRa Packet Forwarder*. [Online]. Available: https://github.com/Lora-net/packet_forwarder

[49] P. Gotthard. (2019). *LoRaWAN Server*. [Online]. Available: https://github.com/gotthardp/lorawan-server

[50] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris, "Link-level measurements from an 802.11b mesh network," in *Proc. Conf. Appl., Technol., Architectures, Protocols Comput. Commun. (SIGCOMM)*, 2004.

[51] imec IDLab. (2017). *LoRaWAN NS-3 Platform*. [Online]. Available: https://github.com/imec-idlab/ns-3-dev-git/tree/lorawan

[52] A. Aimi, S. Rovedakis, F. Guillemin, and S. Secci, "ELoRa: End-to-end emulation of massive IoT LoRaWAN infrastructures," in *Proc. NOMS - IEEE/IFIP Netw. Operations Manage. Symp.*, May 2023, pp. 1–3.

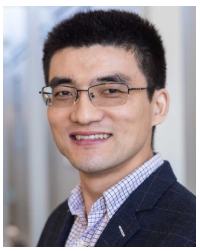
[53] K. Yang, X. Zhao, J. Zou, and W. Du, "ATPP: A mobile app prediction system based on deep marked temporal point processes," in *Proc. 17th Int. Conf. Distrib. Comput. Sensor Syst. (DCOSS)*, Jul. 2021, pp. 83–91.

[54] L. T. DeCarlo, "On the meaning and use of kurtosis," *Psychol. Methods*, vol. 2, no. 3, p. 292, 1997.

[55] K. Yang, X. Zhao, J. Zou, and W. Du, "ATPP: A mobile app prediction system based on deep marked temporal point processes," *ACM Trans. Sensor Netw.*, vol. 19, no. 3, pp. 1–24, 2023.

Kang Yang received the B.E. degree in automation engineering from the School of Electrical and Control Engineering, Xi'an University of Science and Technology, Xi'an, China, in 2016, and the M.E. degree in control engineering from the School of Electronic and Information, Xi'an Jiaotong University, Xi'an, in 2019. He is currently pursuing the Ph.D. degree with the University of California, Merced. His research interests include the AI for wireless networking and sensing.

Miaomiao Liu received the B.S. and M.S. degrees in software engineering from Northeastern University, China, in 2014 and 2018, respectively, and the Ph.D. degree in computer science from the University of California, Merced, USA. Her research interests include wearable-based sensing, video analytics, mobile/edge computing, on-device deep learning, and generative AI.



Wan Du (Member, IEEE) received the B.E. and M.S. degrees in electrical engineering from Beihang University, China, in 2005 and 2008, respectively, and the Ph.D. degree in electronics from the University of Lyon (École Centrale de Lyon), France, in 2011. He was a Research Fellow with Nanyang Technological University, Singapore, from 2012 to 2017. He is currently an Assistant Professor with the University of California, Merced. His research interests include the Internet of Things, distributed networking systems, and mobile computing.