Exploiting Rateless Codes and Cross-layer Optimization for Low-power Wide-area Networks

JIAMEI LV, GONGLONG CHEN, and WEI DONG, Zhejiang University and Alibaba-Zhejiang University Joint Institute of Frontier Technologies, China

Long communication range and low energy consumption are the two most important design goals of Low-power Wide-area Networks (LPWANs); however, many prior works have revealed that the performance of LPWAN in practical scenarios is not satisfactory. Although there are PHY-layer and link-layer approaches proposed to improve the performance of LPWAN, they either rely heavily on the hardware modifications or suffer from low data recovery capability, especially with bursty packet loss patterns. In this article, we propose a practical system, eLoRa, for COTS devices. eLoRa utilizes rateless codes and joint decoding with multiple gateways to extend the communication range and lifetime of LoRaWAN. To further improve the performance of LoRaWAN, eLoRa optimizes parameters of the PHY-layer (e.g., spreading factor) and the link layer (e.g, block length). We implement eLoRa on COTS LoRa devices and conduct extensive experiments on an outdoor testbed to evaluate the effectiveness of eLoRa. Results show that eLoRa can effectively improve the communication range of DaRe and LoRaWAN by 43.2% and 55.7% with a packet reception ratio higher than 60%, and increase the expected lifetime of DaRe and LoRaWAN by 18.3% and 46.6%.

CCS Concepts: • **Networks** → **Network protocol design**; **Network design principles;**

Additional Key Words and Phrases: Internet of Things, LoRa, rateless code

ACM Reference format:
Jiamei Lv, Gonglong Chen, and Wei Dong. 2022. Exploiting Rateless Codes and Cross-layer Optimization for Low-power Wide-area Networks. *ACM Trans. Sensor Netw.* 18, 4, Article 62 (December 2022), 24 pages.
https://doi.org/10.1145/3544560

1 INTRODUCTION

Low-power Wide-area Networks (LPWANs) are emerging communication technologies. They offer wireless communications over a long range and have power and cost advantages over other traditional wireless networks. Among many LPWANs (e.g., LoRaWAN [1], NB-IoT [2], SigFox [3], etc.), LoRaWAN is a technology that has attracted much research interest [4–11].

Long communication range and low energy consumption are the two most important design goals of LoRaWAN. However, many prior works [4, 5, 12, 13] have revealed that the performance

This work is supported by NSFC under grant no. 62072396, Zhejiang Provincial Natural Science Foundation for Distinguished Young Scholars under grant no. LR19F020001, the Fundamental Research Funds for the Central Universities (no. 226-2022-00087), and Alibaba-Zhejiang University Joint Institute of Frontier Technologies.
Authors' address: J. Lv, G. Chen, and W. Dong (corresponding author), Zhejiang University and Alibaba-Zhejiang University Joint Institute of Frontier Technologies, China, 38th, Zheda Rd., Hangzhou, Zhejiang, 310000; emails: lvjm@zju.edu.cn, desword@zju.edu.cn, dongw@zju.edu.cn.

1550-4859/2022/12-ART62 $15.00
https://doi.org/10.1145/3544560

gap of LoRaWAN between the practical scenarios and the theory is still very large. For example, Augustin et al. [13] and Dongare et al. [4] have shown that in a typical outdoor deployment of LoRaWAN, the effective communication range is only 650m (with packet reception ratio higher than 60%).

Recently, many research works have been proposed to improve the performance of LoRaWAN. The PHY-layer approach, e.g., Charm [4], Choir [5], extends the communication range by improving the packet reception ratio based on LoRa signal characteristics. Nevertheless, they rely heavily on hardware modifications to perform sophisticated signal processing, which is not directly accessible on **Commercial-Off-the-Shelf (COTS)** devices. The application-layer approach, e.g., DaRe [6], provides data recovery in LoRaWAN based on the **forward error correction code (FEC)**. However, the error correction capability is highly related to the packet loss patterns (see Section 3). For example, given the fixed coding rate, the consecutive packet loss pattern would easily exceed the correction capability of DaRe, resulting in large retransmission overhead and energy consumption.

To address the above limitations, we propose a practical system, eLoRa, for COTS devices. eLoRa has two important features: (1) eLoRa exploits rateless codes that decouple error recovering units from communication units; i.e., eLoRa transmits large packets that contain multiple blocks with configurable block lengths. eLoRa performs rateless codes on blocks to cope with errors in unreliable links. The use of rateless codes allows decoding with multiple gateways, resulting in a longer communication range and lifetime of LoRaWAN. (2) eLoRa jointly optimizes the parameters of the PHY-layer (e.g., spreading factor) and the link layer (e.g., block length) to further improve the performance of LoRaWAN. eLoRa carefully models the cross-layer parameters that have the most impact on LoRaWAN. For example, the **spreading factor (SF)** indicates how many chips are spreading out for one symbol (i.e., 2^{SF} chips are encoded as SF bits for one symbol). A larger SF denotes that more chips are encoded for one symbol, and thus increases the transmission reliability. However, it lowers the data rate and increases energy consumption. eLoRa provides an optimization framework to optimize the cross-layer parameters.

We implement eLoRa on Dragino LoRa Shield [14] and Dragino LG01 gateway [14] and evaluate its performance in different environments. Results show that eLoRa achieves a highly accurate network model (e.g., the absolute error of 0.98% for reliability estimation). To demonstrate the effectiveness of eLoRa in real scenarios, we implement eLoRa in a real-deployed testbed. Results show that eLoRa can effectively improve the communication range of DaRe and LoRaWAN by 43.2% and 55.7% with a packet reception ratio higher than 60%, and increase the expected lifetime of DaRe and LoRaWAN by 18.3% and 46.6%.

The contributions of this article are summarized as follows:

- We design a practical system, eLoRa, for COTS devices to extend the communication range and lifetime of LoRaWAN. eLoRa provides two application-level parameters that can be specified by network operators (e.g., communication range and lifetime). eLoRa can automatically optimize the parameters based on the monitored network states.
- We jointly consider the parameters of the PHY-layer (e.g., spreading factor) and the link layer (e.g., block length) and propose a cross-layer optimization framework. eLoRa utilizes rateless codes and joint decoding with multiple gateways to improve the performance of LoRaWAN without hardware modifications.
- We implement and evaluate eLoRa on COTS devices. Extensive experiments in the real-world testbed show that eLoRa outperforms the state of the art in terms of lifetime and communication range.

The rest of this article is organized as follows. Section 2 introduces the related work and background of LoRaWAN. Section 3 presents two motivating examples. Section 4 shows the design

of eLoRa. Section 6 presents the evaluation results. Section 7 concludes this article and discusses future research directions.

2 RELATED WORK

LoRaWAN. LoRaWAN uses **chirp spread spectrum (CSS)** for transmitting data considering the requirements of low power, hardware simplicity, and robustness under multi-path and narrow-band interference [1]. LoRaWAN transceivers can operate between 137MHz and 1,020MHz with licensed bands included; however, they are often deployed in ISM bands (e.g., China: 779MHz and 433MHz, EU: 868MHz and 433MHz). The LoRa physical layer may be used with any MAC layer; however, LoRaWAN is the currently proposed MAC. LoRaWAN operates in a simple star topology. To improve communication efficiency, LoRaWAN provides an **adaptive data rate (ADR)** mechanism [1]. Within the maximum retransmission times, LoRaWAN gradually reduces the data rate from 11kbps to 0.25kbps as the retransmission time increases.

Performance Optimization for LoRaWAN. Charm [4] enhances the coverage of LoRaWAN and the battery life of client devices through multiple gateway combinations. It exploits the observation that the weak signals from clients can be identified through filtering the signal patterns of LoRa modulation. Then by coherently combining weak signals received across multiple gateways, the underlying data can be decoded successfully with high probability. AdapLoRa [15] maximizes the network lifetime of LoRa networks by periodically adapting the resource allocation. It develops a network model to capture the link quality variations and network interference and improves the network lifetime by periodically estimating network lifetime with different resource allocations by considering the adaptation overhead (e.g., energy consumed by end-devices to receive the configuration commands). Mu et al. [16] present a low-cost LoRa-based wireless network that collects real-time data from six shuttles circling the university campus. They develop a runtime SF control solution that employs the **K-Nearest Neighbors (KNN)** algorithm to adapt the SF configuration based on the current link condition to increase the data collection throughput while meeting the application reliability. Fahmidayx et al. [17] propose a link-layer protocol to achieve a long-lived LoRa network that enables the nodes with depleting batteries to exploit the superfluous energy of the neighboring nodes with affluent batteries by letting a depleting node offload its packets to an affluent node. Martin et al. [8] develop a link-probing-based approach to automatically adjust the parameters for LoRa transmissions. The parameter selection is purely based on the empirical study, which may not perform well in practical scenarios. LMAC [18] is an efficient **carrier-sense multiple access (CSMA)** protocol for LoRa networks. The authors design three versions of LMAC that respectively implement CSMA for LoRa networks and balance loads of the channels defined by frequencies and spreading factors by using the end nodesâĂŹ local information only and the additional gatewayâĂŹs global information. LMAC brings significant performance improvements in terms of PRR and goodput. The above approaches can be directly applied in eLoRa to further improve the performance and therefore eLoRa is orthogonal to them.

LoRaSim [10] is a simulator that captures low-level LoRa communication behaviors such as capture effect and packet collisions. Floris et al. [11] propose another simulator to study the packet delivery ratio in a large-scale simulated LoRaWAN. Different from the above two simulators that only empirically study the impact of parameters on LoRaWAN's performance, our system not only models LoRaWAN in a cross-layer manner but also provides a joint optimization scheme to improve the transmission performance of LoRaWAN.

LoRa decoding. Choir [5] is a collision decoding approach in LoRaWAN exploiting hardware imperfections. It exploits the observation that signals from the two transmitters are likely to experience a small frequency offset due to the hardware imperfections. Using this offset, collided signals can be decoupled and decoded. Thiemo et al. [9] propose to use directional antennae and

multiple gateways to deal with the inter-network interference. Simulation results show that the performance of LoRaWAN can be improved. NELoRa [19] is a neural-enhanced LoRa demodulation method, exploiting the feature abstraction ability of deep learning to support ultra-low SNR LoRa communication. It takes the spectrogram of both amplitude and phase as input and uses a mask-enabled **Deep Neural Network (DNN)** filter that extracts multi-dimension features to capture clean chirp symbols. Then it uses a spectrogram-based DNN decoder to decode these chirp symbols accurately. All the above approaches rely heavily on the hardware modifications on LoRa devices because of the procedures of complex signals at the PHY layer. On the contrary, eLoRa is a software-based system and can be directly deployed in COTS LoRa devices.

DaRe [6] recovers the lost data in LoRaWAN using the redundant information calculated from previous frames. However, given the fixed coding rate, DaRe suffers from low efficiency under different packet loss patterns. For the example shown in Section 3, when bursty packet loss occurs, more retransmissions are needed to recover the lost packets. Different from DaRe, eLoRa utilizes the rateless code (e.g., LT code [20]) that can automatically achieve a proper bit rate for the given link. Note that the decoding delay of eLoRa may be larger than DaRe. However, LoRaWAN applications are usually non-delay sensitive. For example, the uplink transmission limitation of the LoRaWAN application is 30 seconds on-air time per day per device [21], indicating that we should carefully design the packet recovery mechanism for LoRaWAN applications such that LoRaWAN packets can be successfully decoded at the receiver side within the very limited on-air time. Compared with aggressively dealing with more packets with a lower packet reception ratio by DaRe, it may be more appropriate to carefully recover enough packets with acceptable delay.

Performance Optimization for wireless sensor networks (WSNs). LoRaCP [7] reduces collisions in WSN by leveraging LoRa's capability to transmit control messages over one hop out of band. It is an application of LoRa for WSN. We believe that eLoRa can also benefit the transmission efficiency of control messages in LoRaCP. SYNAPSE++ [22] improves the efficiency of data dissemination by optimizing the degree distribution of LT code [20]. DLT [23] provides long-distance transmissions in WSN by parallelizing the Gaussian Elimination decoding of LT code. pTunes [24] is a framework for runtime adaptation of low-power MAC protocol parameters. Using the information about the current network state, pTunes automatically determines optimized MAC parameters whose performance meets the requirements specification. Different from pTunes, which optimizes parameters using a single gateway, eLoRa also combines multiple gateways to further improve the performance of LoRaWAN.

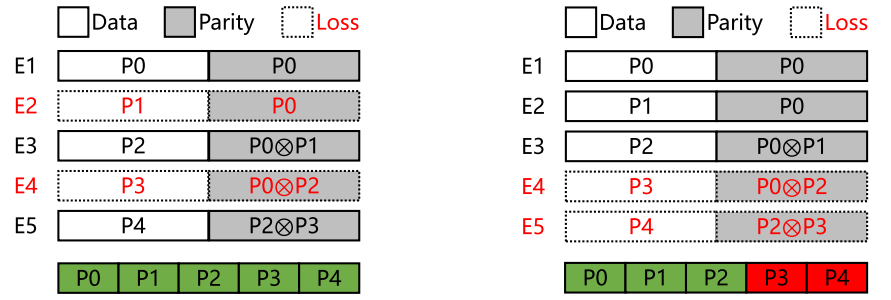
Although eLoRa utilizes several technologies from WSN, eLoRa is significantly different from them in two aspects. First, we perform detailed modeling of the LoRaWAN PHY layer, which is quite different from the WSN PHY layer, e.g., the chirp spread spectrum for LoRa and the direct-sequence spread spectrum for ZigBee. Second, for long-range communications in LoRaWAN, we need to carefully design the system to adapt to the complicated environments, while short-range communications are typical scenarios in WSN.

3 MOTIVATION

In this section, we present an example to show the benefits of using rateless code, and how the performance can be further improved through cross-layer optimization.

3.1 Benefits of Using Rateless Code

DaRe. In certain LoRaWAN data collection scenarios [6], the data sensed by LoRa devices are needed to be transmitted to the gateway in a real-time manner. To this end, Marcelis et al. [6] propose an application-level coding approach, DaRe. It encodes data packets with the redundant



(a) DaRe case 1: P0, P1, P2, P3, P4 are recovered

(b) DaRe case 2: only P0, P1, P2 are recovered

Fig. 1. Motivating example of DaRe. Given five data packets with 40 bytes each to be transmitted, two DaRe encoded packets are lost. Dashed boxes are lost packets; solid boxes are correct packets.

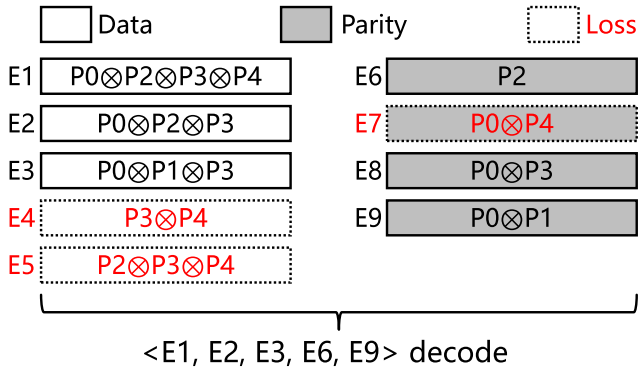


Fig. 2. Motivating example of LT code. Given five data packets with 40 bytes each to be transmitted, three LT encoded packets are lost. The five data packets can be recovered from encoded packets: E1, E2, E3, E6, and E9. Dashed boxes are lost packets; solid boxes are correct packets.

information that is calculated from previous data packets. When encoded packets are lost, DaRe tries to recover the lost data packets using the redundant information.

However, it is possible that DaRe is unable to recover the data packets given the fixed coding rate, resulting in large retransmission overhead. For example, suppose there are five data packets with 40 bytes each to be transmitted to the gateway, then a typical encoding pattern of DaRe packets, as shown in Figure 1. A DaRe encoded packet (e.g., E4) is concatenated with a data packet (e.g., P3) and a parity packet (e.g., $P0 \bigotimes P2$), resulting in the coding rate $R = 0.5$. The parity packet is calculated by randomly XORing two data packets (e.g., degree $d = 0.66$) from the previous three data packets (e.g., sliding window $W = 3$). Although there are two DaRe encoded packets lost (e.g., $PRR = 0.6$), DaRe can still recover the lost data packets (e.g., P1 and P3) from the correctly received packets. For example, P1 can be recovered from E1 and E3, while P3 can be derived from E3 and E5. Therefore, for the packet loss pattern, as shown in Figure 2(a), to reliably transmit $5 \times 40 = 200$ data bytes, $10 \times 40 = 400$ bytes are needed.

However, given the manually configured coding rate, when other packet loss patterns occur (e.g., consecutive lost packets), additional packets are needed. For example, when packets E4 and E5 are lost (e.g., the consecutive packet loss pattern), data packets P3 and P4 cannot be recovered from the redundancy and more redundant packets are needed (e.g., $2/0.6 \approx 3$ extra retransmission packets). Therefore, $10 \times 40 + 3 \times 40 = 520$ bytes are needed.

Rateless code. For a given link, the rateless code can automatically achieve a proper bit rate. To this end, a packet is first split into multiple blocks (say k data blocks). The rateless code then encodes the block with d data blocks, which are randomly chosen from the k data blocks. When receiving an encoded block successfully, the receiver can get an index vector I of data blocks that are encoded. The vectors are accumulated to form a matrix G. The k data blocks can be recovered using **Gaussian Elimination (GE)** once the rank of G is k [25]. By carefully designing the distribution of d, it is highly possible to using $k + \delta$ correctly received blocks to recover k data blocks [25], where δ can be extremely small.

Among the existing lightweight rateless codes [20, 25, 26], we choose LT code [20] because of the lightweight coding operation (e.g., XOR) and the high decoding efficiency optimized by prior works [22, 23]. Figure 2(b) presents an example of using LT code to encode the five data packets (40 bytes each) to be transmitted to the gateway. Note that we treat the packet as the encoded block for illustration purposes. Although three packets are lost (e.g., $PRR \approx 0.667$), the LT code can still recover the five data packets by accumulating six encoded packets. In total, $(3 + 6) \times 40 = 360$ bytes are needed to reliably transmit $5 \times 40 = 200$ data bytes. Compared with DaRe, LT code can significantly reduce the transmitted bytes from 520 bytes to 360 bytes at most.

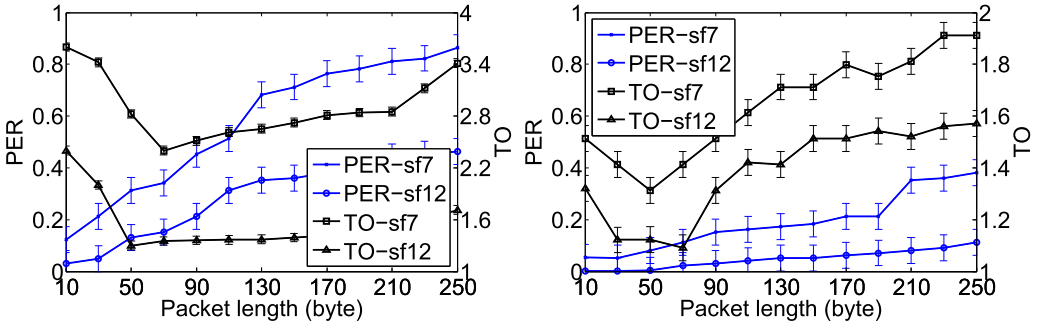
Therefore, we can achieve more efficient transmission by automatically achieving a proper bit rate with LT code, while DaRe relies heavily on manual configurations, which may result in large transmission overhead. Note that the latency of DaRe may be lower than LT code (e.g., LT code needs to wait until all five data packets are ready to perform encoding). However, in most LoRaWAN scenarios [21] that are insensitive to the latency due to the low duty cycle limitations (e.g., <1%), LT code is more suitable than DaRe.

3.2 Benefits of Parameter Optimization

Now that we have improved the error correction capability through LT code, we investigate how to further improve the performance by optimizing the cross-layer parameters. We first conduct empirical experiments to study the impact on LoRa transmissions without any codes considering two representative parameters from the PHY-layer (e.g., spreading factor) and link layer (e.g., block length). The SF denotes the amount of chips per symbol [27]. The larger the SF is, the more reliable the LoRa transmission is, however, resulting in longer packet on-air time. We measure the PER and the normalized transmission overhead (denoted as TO) per useful received byte for each packet length and SF. A low TO value indicates a high goodput [28].

We place an end-device and a gateway at a distance of 100m in an outdoor scenario. The transmission power is varied at the end-device to result in different SNRs (e.g., -8dB and 5dB) at the gateway. We change the value of SF and the packet length while setting other parameters as default in LoRaWAN [1] (e.g., 125kHz bandwidth and 4/5 coding rate). Each experiment is repeated 20 times.

Figure 3 shows the impact of packet length and SF on the PER under different SNRs. We can find that in low SNR scenarios, the packet length has a greater impact on PER. For example, when packet length changes from 80 bytes to 40 bytes, the PER is reduced from 0.4 to 0.3. On the other hand, SF also impacts the PER (e.g., PER is reduced from 0.4 to 0.2 when enlarging SF from 7 to 12). Note that the shortest packet length may not produce the optimal TO due to more overhead incurred from the packet header. Given the same example shown in Figure 2, through the cross-layer parameter optimization (e.g., enlarging SF to 12 and the packet length to 50 bytes), we can reduce the transmitted bytes from 360 to 250. Although we have achieved more efficient transmission, the energy consumption is more due to longer transmission time caused by the larger SF (e.g., LoRa data rate is reduced from 5kbps to 0.2kbps when enlarging SF from 7 to 12). Similarly, a smaller block length leads to more reliable transmission, however, resulting in a larger CRC overhead



(a) Low SNR scenarios (-8dB). (b) High SNR scenarios (5dB).

Fig. 3. Impact of parameters on LoRa performance.

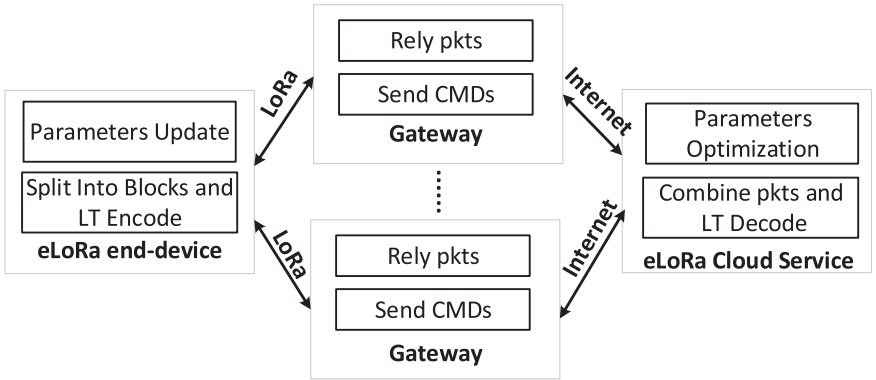


Fig. 4. Overview of eLoRa architecture.

because the number of blocks is increased. Therefore, we need to design a cross-layer optimization framework that jointly considers the high-level requirements (e.g., lifetime, communication range, and latency).

4 DESIGN

In this section, we present overall procedures, design details, and the network model of eLoRa.

4.1 Overview

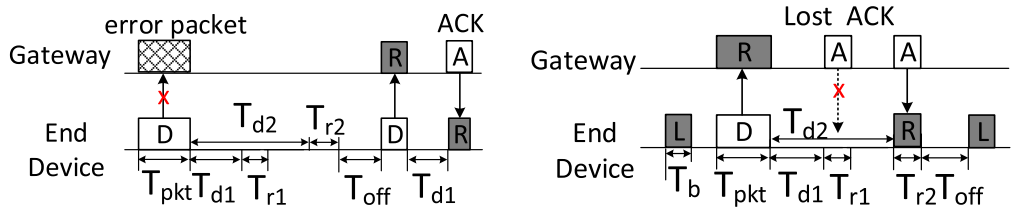
Figure 4 presents the overview of eLoRa architecture. **At the cloud side,** it receives the same packets from multiple gateways to assemble a new packet that contains the most correct blocks. Then the packet is recovered using LT code with high probability. eLoRa provides an attractive feature that network operators can define application-level requirements (e.g., lifetime and communication range). eLoRa will periodically check whether the requirements are violated and automatically carry out the cross-layer parameter optimization based on the network model and the monitored network states. The optimized parameters are then transmitted to the specific end-device over the gateway. **At the gateway side,** it relays the received data to the cloud via the Internet or transmits the parameter settings to the end-device over LoRa links. **At the end-device side,** when receiving a packet from the network layer, it splits the packet into blocks and performs LT coding over the blocks using the optimized parameters. The end-device also updates the parameters (e.g., block size x) upon the commands from the gateway.

ALGORITHM 1: Retransmission protocol at the end-device

```
1  T_bf is the back off time;
2  α is used to determine the number of merged packets;
3  case ACK received do
4      clear up pkt in the sender buffer sbf;
5      state = next_pkt;
6  case NAK received do
7      extract the num. of blocks CB from NAK;
8      generate CB blocks using LT code;
9      send the encoded pkt after T_bf and start ACK timer;
10 case ACK timer times out do
11     retransmit pkt after T_bf;
12 case pkt received from network layer do
13     put pkt into the buffer bf;
14     if len(sbf) ≥ len(pkt)*α and state == next_pkt then
15         generate len(sbf)/x data blocks with x bytes each;
16         generate len(sbf)/x encoded blocks using LT code;
17         send the pkt with encoded blocks and start ACK timer;
18         state = send;
19     else
20         move pkt from bf into the send buffer sbf;
```

ALGORITHM 2: Retransmission protocol at the cloud side

```
1  G_n is the number of gateways receiving pkts from end-device n;
2  M is the communication distance; L is the latency;
3  case pkt received from gateway do
4      put pkt into the receive buffer rbf;
5      if G_n packets are received then
6          combining most correct blocks in pkt;
7          pkt2 = lt_decode(pkt);
8          if pkt2 is correct then
9              reply ACK and deliver pkt2 to the network layer;
10         else
11             reply NAK with the num. of incorrect blocks CB;
12 case constraint violation do
13     find optimized parameters;
14     if no solution then
15         decrease M or increase L until find the solution;
16     else
17         send parameter settings;
```



(a) LoRaWAN Class A. (b) LoRaWAN Class B.

Fig. 5. Two typical LoRaWAN classes.

4.2 Key Procedures at the End-device and the Cloud

Algorithm 1 shows the key procedures at the end-device side. To improve the transmission efficiency, eLoRa will first accumulate the packets received from the network layer into a send buffer sbf until it is large enough to be sent (e.g., packet size is at least larger than $\alpha * len(pkt)$), where α can be used to determine the number of merged packets. Then eLoRa splits the packet into multiple blocks to perform the LT encode. When there is a NAK received, eLoRa will generate CB new LT encoded blocks for retransmissions.

Algorithm 2 presents the key procedures on the cloud side. The network operator can adjust four high-level parameters: the communication range M, the lifetime T, the reliability R, and the latency L. M or T can be selected as the optimization goal, while others are treated as the constraints. eLoRa periodically checks the network states and performs cross-layer optimizations when constraints are not satisfied. When there are no proper solutions under current network states, to best fit the requirements, eLoRa gradually relaxes the constraints until the optimized parameters are found (e.g., decreasing M or increasing L). To perform LT decoding, eLoRa infers a seed using a similar way in [23] as from the packet id and the offset of the block.

4.3 Network Model and Basic Notations

We consider a typical LoRaWAN network in that the end-device transmits packets to the gateway through one-hop [21]. There are three classes defined in LoRaWAN [1], i.e., Class A, Class B, and Class C. By default, all three classes should implement the basic procedure of Class A. Class A and Class B are low-power MAC protocols, enabling duty cycle to save the end-device's energy. eLoRa builds on two representative low-power MAC protocols of LoRaWAN, i.e., Class A and Class B [1]. Note that we do not model the Class C because it keeps the end-device's radio on and consumes too much energy, which is rarely used in practical scenarios.

Figure 5(a) shows the basic procedure of LoRaWAN Class A. For the end-device, it wakes up every T and sends packets to the gateway (i.e., the uplink transmission) if any. Once the uplink transmission is done, the end-device must wait a specific time T_{d1} and then open two short receive windows (i.e., windows length T_{r1} and T_{r2}) for the potential packets from the gateway (i.e., the downlink transmission). The interval between two short receive windows is $T_{d2} - T_{d1}$, as shown in Figure 5(a). The length of the first window is fixed (i.e., set to 1 second by default), whereas the length of the second window can be modified by sending MAC commands by the gateway. If there is no downlink transmission at any two of the receive windows, the end-device will go to sleep for T_{off}. When a gateway wants to send downlink packets, it must keep on listening until an uplink packet is received correctly. The gateway can send multiple downlink packets by setting the FPending bit to one in the packet header, while the end-device can only send or resend one packet every T. T_{pkt} denotes the time for transmitting a packet, and T_{ack} denotes the time for transmitting an ACK.

Table 1. Notations Used in This Article

Notation	Meaning
n	The current end-device
S	The set of end-devices in the network
G_n	The number of gateways that can receive packets from the end-device n
H	Header size in one packet
T_{d1}, T_{d2}, T_{r1}, T_{r2}, T, T_{off}, T_{la}, T_r	Class A parameters (see Figure 5(a) and Section 4.5)
T_{d1}, T_{d2}, T_{r1}, T_{r2}, T, T_{off}, T_{lb}, T_b, T_r	Class B parameters (see Figure 5(b) and Section 4.5)
U_{tn}	Useful data rate at the end-device n (bytes/h)
x_n	Block length at the end-device n
d_{nj}	The distance between the gateway j and the end-device n
$T_{pkt}(x)$	Time to transmit one packet with x bytes
T_{ack}	Time to transmit one ACK
s, c, b, P_t	LoRa PHY-layer parameters, spreading factor (s), coding rate (c), bandwidth (b), transmission power (P_t)
α	The number of data packets needed to be merged for the LT encode
$\boldsymbol{v_n}$	Parameters to be optimized and used at the end-device n, $\{s, c, b, P_t, x_n, \alpha\}$
N	The maximum retransmission time
F_{tnb}	The block transmission rate at end-device n
F_{tnp}	The packet transmission rate at end-device n

Figure 5(b) shows the basic procedure of LoRaWAN Class B. In addition to the two receive windows after the uplink transmission as defined in Class A, the end-device in Class B also opens another listening window lasting for T_b every T to receive downlink packets. The extra window is synchronized by the gateway with a factor of T time.

There are mainly four runtime-adjustable parameters in the LoRaWAN PHY-layer: spreading factor s (SF), coding rate c (CR), bandwidth b (BW), and transmission power P_t [27]. SF determines how many chips are spreading out in a given bandwidth for one symbol (i.e., 2^{SF} chips are encoded as SF bits for one symbol). It can be set between 7 and 12. A higher SF denotes more chips are encoded for one symbol, and thus increases the receiver sensitivity and the range of the signal. However, it lowers the data rate and therefore increases the transmission duration and energy consumption. BW is the width of frequencies in the transmission band. Higher BW gives a higher data rate (thus shorter time on air). In a typical LoRa deployment, BW can be set to 125kHz, 250kHz, or 500kHz. CR is the amount of FEC (e.g., Hamming code [27]) that is applied to the message to protect it against interference. Higher CR makes the message longer and therefore increases the time on-air. Transmission power can be varied from 5dBm to 23dBm (on rf95 [29]).

Table 1 summarizes the notations we use to denote network states and protocol-dependent quantities.

4.4 Application-level Metrics

In a typical data collection scenario with static LoRa devices, we simultaneously consider two important application-level metrics of real-world LoRaWAN applications [21]: lifetime T, communication range M.

Lifetime. Similar to prior works [28, 30], we first analyze the lifetime in terms of its duty cycle, i.e., the fraction of time that is used in transmission mode (D_{tn}) and receiving mode (D_{rn}). For the Dragino LoRa Shield [29], the current of radio in transmission mode is 120mA (with transmission power at 23dBm) and in receiving mode is 16.6mA, while the current of MCU is only 3.5mA. Then the duty cycle of node n is calculated as $D_n = D_{tn} + D_{rn}$. We want to minimize D_n to meet the application-level requirements set by the network operator. Given a battery capacity Q, we define the node lifetime T_n of end-device n as [24]

$$T_n = Q/(D_{tn} \cdot I_{tx} + D_{rn} \cdot I_{rx} + D_{in} \cdot I_i), \tag{1}$$

where I_{tx}, I_{rx}, and I_i are the current draws of the radio in transmitting, receiving, and idle mode. $D_{in} = 1 - D_{tn} - D_{rn}$.

Communication range. We let M_n denote the distance between the end-device n and the gateway, and R_n denote the packet reception ratio of a gateway for the end-device n. The distance of every end-device is recorded at the initialization step at the gateway. The network operator gives the minimum communication distance d; our system would try to ensure the PRR of all nodes within the range d is higher than the PRR requirement r. It has a default value (e.g., 60% [12]) in our system and can be adjusted by the network operator.

Then we can express the optimization problem in our system in the following general form:

$$\begin{aligned} \min/\max \quad & A_1(c) \\ s.t. \quad & A_2(c) <, > C_1, \end{aligned} \tag{2}$$

where A_i is one among $\{T_n, M_n, R_n\}$ and C_i is the requirement. We also consider the transmission latency L_n in the constraint. For example, given the PRR requirement r, the lifetime requirement t, and the latency requirement l, the optimization problem is

$$\begin{aligned} \max \quad & M_n \\ s.t. \quad & T_n > t, R_n > r, L_n < l, n \in S. \end{aligned} \tag{3}$$

Note that to maximize the communication range, we actually first transform the above problem into: maximize the R_n while meeting $T_n < t$ and $L_n < l$. After optimization, we choose the end-device n with the maximum distance from all end-devices satisfying $R_n > r$ as the optimized communication range. In the following, we present the details of modeling the application-level metrics and relevant requirements.

4.5 Metric Modeling

We define the optimization parameters as the set $\boldsymbol{v_n}$, which consists of the number of data packets α to be merged, the block size x, and all other PHY-layer parameters (e.g., spreading factor s, coding rate c, bandwidth b, and transmission power P_t). Then we present the relationship among $\boldsymbol{v_n}$ and high-level metrics, i.e., lifetime T, reliability R, and latency L.

Lifetime modeling. To model the lifetime of the LoRa end-device, we first need to accurately model the duty cycle, i.e., $D_n = D_{tn} + D_{rn}$. As shown in Figure 5, the communication procedures of LoRaWAN classes are different, resulting in different duty cycle modeling.

For **LoRaWAN Class A**, it consists of the time used in transmitting packets (D_{tn}) and receiving ACK/NAK from the gateway (D_{rn}). When ACK/NAK is not received, the end-device needs to listen

for T_{d2} and T_{r2}:

$$D_{na} = D_{tn} + (1 - D_{rn})\frac{T_{d2} + T_{r2}}{T} + D_{rn}\frac{(T_{d1} + 2 \cdot T_r + T_{d2})}{2 \cdot T}, \tag{4}$$

where $T = T_{pkt} + T_{off} + T_{la}$. $T_{la} = T_{d2} + T_{r2}$ when there is no ACK/NAK received. On the other hand, we assume that the probability of successfully receiving ACK/NAK in the first receiving window T_{r1} and the second receiving window T_{r2} is equal, and T_r is the time for receiving ACK/NAK from the gateway; then $T_{la} = 0.5 \cdot (T_{d1} + T_r) + 0.5 \cdot (T_{d2} + T_r)$. D_{tn} can be further divided into the time fraction in transmitting LT encoded blocks and packet headers:

$$D_{tn} = F_{tnb} T_{pkt}(x_n) + F_{tnp} T_{pkt}(H), \tag{5}$$

where F_{tnb} is the rate of transmitting blocks (given in Equation (11)), and F_{tnp} is the rate of transmitting packets that may contain multiple blocks (given in Equation (19)). D_{rn} consists of the time fraction for receiving ACK/NAK from the gateway:

$$D_{rn} = F_{tnp} PRR_{ack}(\boldsymbol{v_n}) T_{ack}. \tag{6}$$

For **LoRaWAN Class B**, its difference from Class A is the additional listening window T_b for receiving packets from the gateway:

$$D_{nb} = D_{tn} + (1 - D_{na})\frac{T_b}{T} + D_{na}, \tag{7}$$

where $T = T_{pkt} + T_{off} + T_{la} + T_{lb}$, where T_{la} is the same with class A. $T_{lb} = T_b$ when there are no packets received, and otherwise T_{lb} is the total time for successfully receiving packets from the gateway.

Reliability modeling. It is expressed as the packet reception ratio from the end-device n to the gateway j. It depends on parameter vector $\boldsymbol{v_n}$:

$$R_n = \frac{\alpha \cdot U_{tn}}{x_n \cdot N_{tnb}(\boldsymbol{v_n})}, \tag{8}$$

where U_{tn} denotes that there is one data packet with U_{tn} bytes generated per hour, and $N_{tnb}(\boldsymbol{v_n})$ denotes the expected time of retransmitting LT encoded blocks.

Latency modeling. We define latency as the time needed for the gateway to successfully receive a packet from the end-devices. When packet merging is enabled (e.g., $\alpha > 1$), the latency is defined as the time for successfully receiving the first generated packet. Then the latency consists of the time for transmitting or retransmitting packets, and the time for waiting new packets and backoff T_{wait}:

$$L_n = N_{tnb} T_{pkt}(x) + N_{tnp} T_{pkt}(H) + (N_{tnp} - 1) \cdot T_{wait}, \tag{9}$$

where N_{tnp} is the expected number of transmitting packets that may contain multiple LT encoded blocks (given in Equation (23)). T_{wait} for LoRaWAN Class A consists of the time in listening when no ACK/NAK is received, waiting for new packets, and backoff:

$$T_{waita} = (1 - PRR_{ack})(T_{d2} + T_{r2}) + T_{bf} + \alpha \cdot T_D, \tag{10}$$

where T_D is the time for generating a new packet from the network layer. For LoRaWAN Class B, there are additional listening windows opened for T_b and therefore $T_{waitb} = T_{waita} + T_b$.

4.6 Link Layer and PHY-layer Modeling

In this subsection, we will present the building blocks for modeling the previous metrics.

Calculating F_{tnb}. We denote F_{tnb} as the block transmission rate of an end-device that depends on the block length x and the number of merged packets α:

$$F_{tnb} = N_{tnb}(\boldsymbol{v_n}, G_n) \cdot \alpha \cdot U_{tn}/x_n, \tag{11}$$

where $N_{tnb}(\boldsymbol{v_n}, G_n)$ is the expected number of block transmissions given the parameter vector $\boldsymbol{v_n}$ and the number of gateways G_n, so that all data blocks (i.e., $\alpha \cdot U_{tn}/x_n$ blocks) can be decoded successfully on the cloud.

To illustrate $N_{tnb}(\boldsymbol{v_n}, G_n)$ more clearly, we revisit the process of transmitting data with eLoRa. Assuming there are k blocks, during transmission to the cloud, some blocks may be lost or corrupt due to the poor link quality. When they reach the cloud, some blocks may not be able to be decoded due to the lack of information. Therefore, we consider the loss during transmission and decoding when estimating $N_{tnb}(\boldsymbol{v_n}, G_n)$. Based on the above, $N_{tnb}(\boldsymbol{v_n})$ can be presented as

$$N_{tnb}(\boldsymbol{v_n}, G_n) = \frac{1}{BRR_g(\boldsymbol{v_n}, G_n) \cdot \frac{\sum_{m=0}^{M} BDR_{lt}(m,k)}{M}}, \tag{12}$$

where $BRR_g(\boldsymbol{v_n}, G_n)$ is the block reception ratio of gateways, and $BDR_{lt}(m, k)$ is the block decoding ratio of the cloud server.

$BDR_{lt}(m, k)$ is the probability that the server can successfully decode k data blocks when receiving m blocks from the gateway. M is the maximum number of blocks that the server can receive, which is equal to $k \cdot N \cdot BRR_g(\boldsymbol{v_n}, G_n)$, where N is the maximum retransmission time. Because we utilize the Gaussian Elimination to decode LT encoded blocks, calculating BDR_{lt} turns to calculating the probability of receiving m blocks with k ranks (e.g., m successfully received blocks, k data blocks).

The packets received by the cloud can be seen as an $n \times k$ matrix, where n is the number of packets received by the cloud and can be given by m/G, where G is the average number of blocks encoded per packet (i.e., the degree of LT coding). Calculating BDR_{lt} is actually computing the probability that a random binary matrix is full rank, which can be divided into two steps: (1) calculating the number of matrices with full rank and (2) calculating the probability of full rank. According to [31], the total number of full-rank $n \times k$ matrices can be given by

$$F(n, k) = (2^n - 1)(2^n - 2) \cdots (2^n - 2^{k-1}) = \prod_{i=0}^{k-1} (2^n - 2^i). \tag{13}$$

If every $n \times k$ matrix is equally likely to occur, the probability of selecting a matrix with full rank is

$$P(n, k) = \frac{F(n, k)}{2^{nk}} = (1 - 2^{-n})(1 - 2^{-n+1}) \cdots (1 - 2^{-n+k-1}) = \prod_{i=0}^{k-1} (1 - 2^{i-n}). \tag{14}$$

Based on Equation (14), BDR_{lt} can be presented as

$$\mathrm{BDR}_{lt}(m, k) = \prod_{i=0}^{k-1} (1 - 2^{i-m/G}). \tag{15}$$

BRR_g is defined as when the block is correctly received at any one of the gateways, given the number of gateways G_n that can receive the packets from the end-device n. Then it can be directly

fed into the LT decoder:

$$BRR_g(\boldsymbol{v_n}, G_n) = 1 - \prod_{j=1}^{G_n}(1 - BRR_{raw}(\boldsymbol{v_n}, d_{nj})), \tag{16}$$

where d_{nj} denotes the distance between the end-device n and the gateway j, and $BRR_{raw}(\boldsymbol{v_n}, d_{nj})$ denotes the block reception rate without LT decoding:

$$BRR_{raw}(\boldsymbol{v_n}, d_{nj}) = (1 - B(\boldsymbol{v_n}, d_{nj}))^{8(x+1)}, \tag{17}$$

where $B(\boldsymbol{v_n}, d_{nj})$ is the bit error rate given the PHY-layer parameters [27] and can be expressed as [32]

$$B(\boldsymbol{v_n}, d_{nj}) = Q\left(\frac{\log_{12}(s)}{\sqrt{2}}\frac{E_b}{N_0}\right), \tag{18}$$

where $\frac{E_b}{N_0}$ is the signal-to-noise ratio per bits, and it is related to the SF s, the CR c [27] and the distance d_{nj}, and the transmission power P_t. The relationship can be derived based on the existing path loss model [10].

Calculating F_{tnp}. We denote F_{tnp} as the rate of transmitting packets that may contain multiple LT encoded blocks from an end-device:

$$F_{tnp} = N_{tnp}(\boldsymbol{v_n})/\alpha, \tag{19}$$

where $N_{tnp}(\boldsymbol{v_n})$ denotes the expected number of packet transmissions that may contain multiple LT encoded blocks:

$$N_{tnp}(\boldsymbol{v_n}) = \log_{(1-BRR_{lt}(m,k,\boldsymbol{v_n}))}\left(1 - \frac{N_{tnb}(\boldsymbol{v_n}) \cdot x_n}{PRR_{ack} \cdot \alpha \cdot U_{tn}}\right). \tag{20}$$

Calculating T_{pkt}. The packet transmission time T_{pkt} is given by the product of the number of symbols in the packet $N_{symbols}$ and the time to transmit one symbol $T_{symbols}$. As defined in the LoRa datasheet [33], $N_{symbols}$ can be calculated as

$$N_{symbols} = P + 4.25 + 8 + \max\left(\left\lceil\frac{8l - 4s + 24}{4s}\right\rceil(c + 4), 0\right), \tag{21}$$

where P is the number of symbols in preamble, which defaults to 8 in eLoRa. l is the packet length. s is the spreading factor (6 to 12). DE is the indicator of $LowDataRateOptimize$. When $T_{symbols} \geq$ 16ms, DE = 1; otherwise, DE= 0. c is the coding rate (1 corresponding to 4/5, 4 to 4/8).

$T_{symbols}$ is a function of SF s and BW b and given by

$$T_{symbols} = \frac{2^s}{b}. \tag{22}$$

Based on Equations (21) and (22), T_{pkt} can be calculated as

$$T_{pkt} = \frac{P + 12.25 + \max(\lceil\frac{8l-4s+24}{4s}\rceil(c + 4), 0) \cdot 2^s}{b}. \tag{23}$$

4.7 Cross-layer Optimization

Applying the optimization problem in Equation (2) to Class A and Class B leads to a **mixed-integer nonlinear program (MINLP)** with non-convex objective and constraint functions. To solve it efficiently, we use the ECLiPSe constraint programming system [34]. Its high-level programming paradigm allows for succinct modeling of our optimization problem. We solve the optimization problem at the cloud side and send the parameter update command to the gateway to distribute the settings to the end-devices.

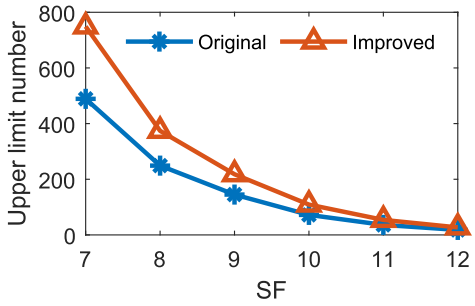


Fig. 6. The upper limit number of supported IoT devices by a gateway.

5 IMPROVED ACK MECHANISM

With eLoRa, the gateway needs to relay ACK/NAK packets from the cloud server to end-devices. However, in LoRaWAN, the gateway is constrained by the duty cycle requirements as well, and when its transmission quota runs out, it cannot transmit any ACK/NAK packets. We build an analytical model to illustrate the upper limit of the number of end nodes supported by a gateway implemented with eLoRa:

$$N = \frac{D_{ug}}{F_{tnp} T_{ack}}, \tag{24}$$

where D_{ug} is the regulated duty cycle of the gateway, F_{tnp} is the rate of transmitting packets, and T_{ack} is the time to transmit one ACK/NAK. When D_{ug} = 1% and the packet transmitting rate is 1 packet/h, the gateway supports up to 1,994 end-devices theoretically with configurations that sf = 7, c = 1, bw = 500kHz. When the configuration is sf = 12, c = 4, bw = 125kHz, the gateway supports the smallest number of devices, i.e., 15 devices. As seen from the model, eLoRa may not be able to scale well for some scenarios where the scalability of the gateway is the bottleneck (e.g., smart metering).

To improve the scalability of eLoRa, we improve the ACK mechanism of eLoRa: (1) Using cumulative acknowledgment instead of transmitting the ACK immediately. The server sends ACK after waiting for a certain period or receiving a certain number of packets unless the packets have been all decoded. A single acknowledgment is in response to a finite number of frames received to reduce the number of ACK packets. (2) Multicasting-based ACK. When there are several ACK packets to be sent, the server can merge these packets into one packet, which will be multicast by the gateway. Multicasting can reduce the air time of LoRa ACK frames by sharing the preamble, header, and so forth. It is noted that multicasting is only allowed for Class B and Class C. (3) The server periodically sends control packets to end-devices, e.g., packets related to parameter update. The server attaches ACK information to these control packets. It is worth emphasizing that (2) may lead to the energy consumption of end-devices increasing due to longer receiving time, so it is suitable for networks where the bottleneck is the number of nodes rather than the lifetime.

We illustrate the upper limit of the end-devices supported by the gateway with the original ACK mechanism and the improved mechanism. The traffic load of each end-device is 80 bytes/h. The coding rate is 4/5. The duty cycle of the gateway is 1%. The results are shown in Figure 6. As can be seen from the figure, the number of devices supported by the gateway has increased by about 150% with the improved ACK mechanism.

6 EVALUATION

In this section, we introduce the testbed setup and present the evaluation results of eLoRa.

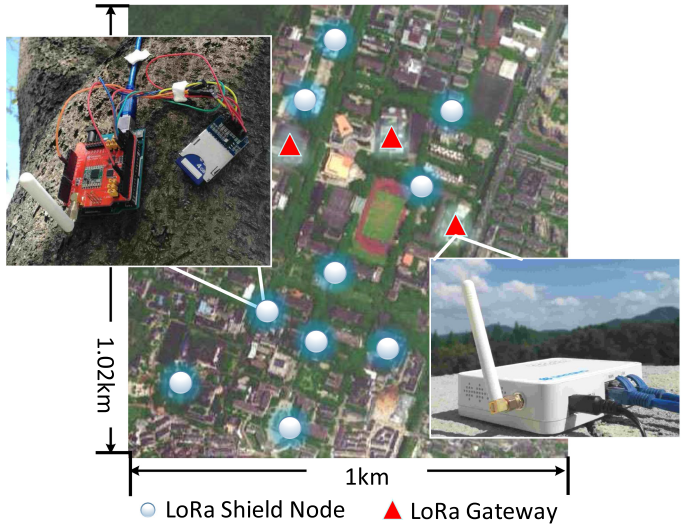


Fig. 7. Outdoor testbed setup. The testded is built in a 1.02km × 1km campus with 10 LoRa nodes and 3 LoRa gateways. The circles in the figure indicate LoRa nodes and the triangles indicate LoRa gateways.

6.1 Experimental Setup and Metrics

Testbed. As shown in Figure 7, to evaluate the performance of eLoRa in practical scenarios, we build a LoRaWAN testbed in a 1.02km × 1km campus with 10 LoRa Shield nodes and three LoRa gateways. Gateways are placed on the roof to achieve wider coverage. Each gateway connects to the Internet via a wired connection, so that we can access the received data through the web application. We combine the data received from multiple gateways to jointly decode the corrupted packets with LT code [22]. Each LoRa client is equipped with a 16,000mAh portable charger and a 4GB TF card to record the necessary information (e.g., transmitted packets) as the ground truth. LoRa clients are placed on the tree.

Metrics. We use the two application-level metrics introduced in Section 4 (e.g., communication range M and lifetime T) to evaluate the overall performance of eLoRa. The communication range is recorded as the longest distance from the end-device to the closet gateway [4] while meeting the reliability requirement R (e.g., $R > 60\%$ [12]). To measure end-device's lifetime, we modify the chip driver (i.e., rf95 [29]) of Dragino LoRa Shield to record the fractions of time the radio is in receiving, transmitting, and idle mode. Then, we compute projected lifetimes using Equation (1) and current draws from the rf95 data sheet [29], assuming batteries constantly supply 2,000mAh at 3V. The lifetime is estimated based on the energy consumption recorded within 1 week, instead of actually running out of the battery.

Requirements. We consider a data collection scenario [21] in LoRaWAN that maximizes the end-device's lifetime while providing a certain communication range.

$$\begin{aligned} \max \quad & T_n \\ s.t. \quad & M_n > 600m, R_n > 60\%, L_n < 5h, n \in S \end{aligned} \tag{25}$$

eLoRa solves Equation (25) at runtime to determine the optimized parameters. If there are no solutions, eLoRa picks the maximized T_n with the other constraints decreased step by step (e.g., decreasing communication range with 100m per step).

Methodology. We compare eLoRa with two existing approaches: (1) the default LoRaWAN [1] and (2) the LoRaWAN with data recovery code DaRe [6]. The parameter settings of default

Table 2. Parameter Settings

Setup	*Class A*		*Class B*		*Specified Range*
	S1	S2	S3	S4	
T_{cc}	1%	10%	5%	15%	-
U_{tgn}(bytes/h)	100	150	100	150	80~260
s	9	7	9	7	7~12
c	4/8	4/5	4/8	4/5	4/5, 4/6, 4/7, 4/8
b(kHz)	250	125	250	125	125~500
P_t(dBm)	20	14	20	14	5~23
G_n	3	1	3	1	1~3

Class A and Class B are two typical device types of LoRaWAN.

LoRaWAN are shown in Table 2. We note that $T_{cc} = T_{pkt}/T$, where T_{pkt} actually denotes the time for transmitting and retransmitting the packets. We implement eLoRa and DaRe as the 2.5 layer on top of the LoRaWAN, respectively. For Class A and Class B, the approaches are denoted as A-DaRe, B-DaRe, A-eLoRa(wG), and B-eLoRa(wG) (parameter settings S1 and S3 in Table 2 are used for Class A and Class B). For DaRe we set the coding rate $R = 0.5$, window size $W = 8$, and degree $d = 0.83$ according to [6]. We craft an eLoRa version without the combinations of multiple gateways, e.g., A-eLoRa(woG) and B-eLoRa(woG).

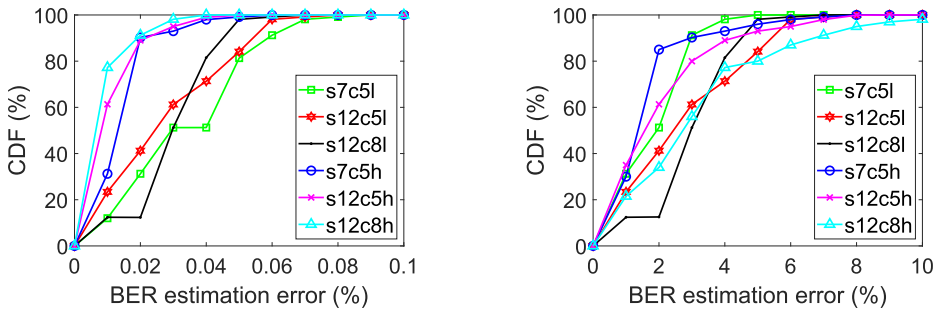
We evaluate the model accuracy in Section 6.2 and the overall performance of eLoRa in Section 6.4. We evaluate the detailed performance of eLoRa under static conditions and dynamic conditions in Section 6.5. We use the parameter settings shown in Table 2, and fix one of the parameters while varying other parameters. We measure the overhead of eLoRa in Section 6.6. We test the runtime of eLoRa when the number of data packets α and the block size x are getting larger. For ease of deploying eLoRa in practical scenarios, we present the impact of constraint settings on the optimization results in Section 6.7.

We use the USRP [35] to generate controllable interference patterns on the sub-1GHz band. To generate dynamic link qualities scenarios, we place the USRP near the gateway to interfere with the packet reception. The interference fraction is computed as the ratio of the interference duration and the total duration (i.e., 100 minutes). A smaller interference fraction indicates a more frequently changing channel. The experiments are conducted 10 times and the results are averaged.

6.2 Model Validation

Figure 8 shows the error of estimating BER in terms of SNR, SF, and CR. The label $sxcy[l, h]$ denotes the parameter settings of SF x and CR y and with low SNR l (−10dB) or high SNR h (2dB). Figure 8(a) shows that the absolute error is 0.25% on average under different SNR, SF, and CR. We note that the estimating error under high SNR (e.g., 0.156%) is a little bit smaller than under low SNR (e.g., 0.35%), because under low SNR the communication link is unreliable and more unpredictable factors may manifest, resulting in a bit higher estimating error. Figure 8(b) shows the relative error. Results show that the 95th-percentile average is 4.53% under different SNR, SF, and CR. We note that the estimating error with SF 12 is a bit larger than with SF 7. The reason is that a higher SF value has smaller BER due to more processing gain and amplifies the bias when calculating the relative error.

Table 3 presents the accuracy of estimating high-level metrics (i.e., reliability R and lifetime T) under different parameter settings shown in Table 2. In addition to the static parameter settings, we also perform each parameter setting with eLoRa enabled. We let eLoRa adapt the cross-layer parameters every 20 minutes. We see that eLoRa achieves high accuracy in reliability (0.98% errors



(a) Absolute error. (b) Relative error.

Fig. 8. Errors of estimating BER. "s$xcy[l, h]$" denotes the parameter settings of SF x, CR y, and with low SNR l (−10dB) or high SNR h (2dB).

Table 3. Absolute Errors of Estimating High-level Metrics

	Class A			***Class B***		
	S1	S2	S1-eLoRa	S3	S4	S3-eLoRa
δ**(R)[%]**	0.14	−0.86	0.21	2.98	−1.25	0.45
δ**(T)[year]**	0.05	0.15	−0.23	0.14	−0.15	0.11
δ**(L)[h]**	0.12	0.14	0.2	−0.05	0.12	0.13

on average), lifetime (0.138 year errors on average), and latency (0.126 hour errors on average) under different parameter settings.

6.3 The Overhead of eLoRa

Low power consumption is essential for resource-constrained IoT end-devices. We measure the overhead of eLoRa in terms of LT encoding and receiving parameters, respectively.

The energy consumption of LT coding can be calculated as

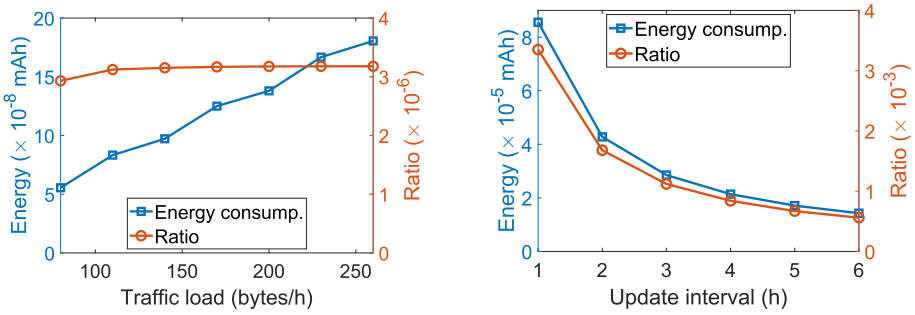
$$E_{coding} = T_{coding} E_{mcu}, \tag{26}$$

where T_{coding} is the time fraction of performing LT coding and E_{mcu} is the current of MCU. T_{coding} is a product of the time to perform one encoding operation and the number of operations. eLoRa uses XOR to encode data; the time to perform one encoding operation is only about one clock cycle (i.e., 62.5ns with Arduino Uno). Therefore, the energy consumption of LT coding is extremely small, as shown in Figure 9(a), and only accounts for about 0.0003% of the total energy consumption with different traffic loads.

The energy consumption of receiving parameters from the cloud server can be presented by

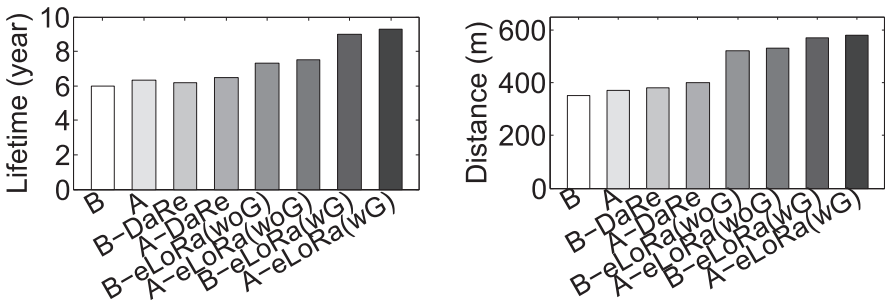
$$T_{para} = T_{para} E_{recv}, \tag{27}$$

where T_{para} is the time fraction of receiving parameters and equals $F_{para} \cdot T_{pkt}(C) \cdot F_{para}$ is the frequency of parameter update and $T_{pkt}(C)$ is the time used in receiving one control packet whose size is C bytes from the cloud server. E_{recv} is the current of radio in receiving mode, which is 16.6mAh (see Section 4.4). Based on the above analytical model, we illustrate the energy consumption of parameter update with different update intervals. The results are shown in Figure 9(b). When the update interval increases, the lifetime decreases slightly. For a static scenario, users can improve the end-devices' lifetime by reducing the frequency of parameter update. For a dynamic scenario, users can adopt the adaptive update mechanism similar to [15, 16] to achieve efficient parameter updates to further improve the lifetime of end-devices.



(a) Overhead of LT coding. (b) Overhead of parameter update.

Fig. 9. Overhead of eLoRa.

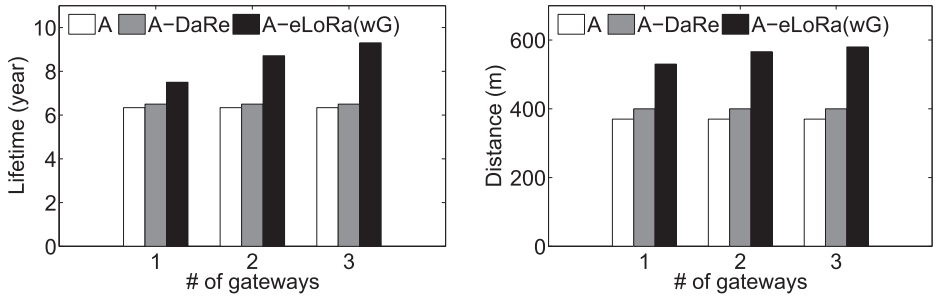


(a) Lifetime. (b) Communication range.

Fig. 10. Overall performance comparison. "*X*-*Y*(*Z*)" denotes different scenarios. *X* is LoRaWAN classes. *Y* indicates the 2.5 layer protocol (i.e., eLoRa or DaRe). *Z* indicates whether there are combinations of multiple gateways.

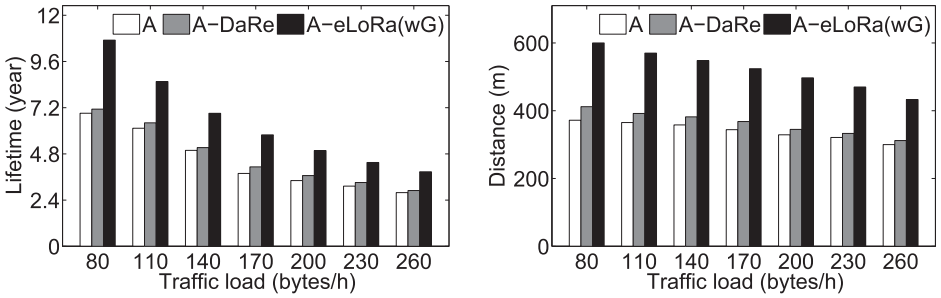
6.4 Overall Performance

Figure 10 shows the overall performance improvements of eLoRa compared with DaRe and LoRaWAN. DaRe and eLoRa are used as the 2.5 layer protocol of LoRaWAN MAC, respectively, and are denoted as A+DaRe, B+DaRe, A+eLoRa/wG, and B+eLoRa/wG. We also craft an eLoRa version without the combinations of multiple gateways, e.g., A+eLoRa/woG and B+eLoRa/woG. Results show that eLoRa achieves the best performance over other approaches for both Class A and Class B. For example, when combining multiple gateways, eLoRa increases the communication range and the expected lifetime of the default LoRaWAN by 55.7% and 46.6% (both are averaged in Class A and Class B). Note that the performance improvement of DaRe for LoRaWAN is subtle (e.g., 8.1% and 2.5% improvements in terms of communication range and lifetime) due to the default conservative parameter settings and the low decoding efficiency in the pattern of bursty packet loss. Different from DaRe, even without the combination of multiple gateways, eLoRa achieves better performance than the default LoRaWAN by 43.2% and 18.3% in terms of the communication range and lifetime, because eLoRa utilizes rateless codes to recover the lost packets under any patterns (unlike DaRe, which is not robust to the bursty packet loss) and further optimizes the performance by adjusting the parameters from both the PHY-layer (e.g., SF) and the link layer (e.g., block length). A-eLoRa(wG) achieves longer lifetime than B-eLoRa(wG) by 3.4% on average because in Class B more idle listening windows are opened for receiving packets from the gateway, resulting in more energy consumption.



(a) Lifetime. (b) Communication range.

Fig. 11. Impact of the number of gateways.



(a) Lifetime. (b) Communication range.

Fig. 12. Impact of traffic load.

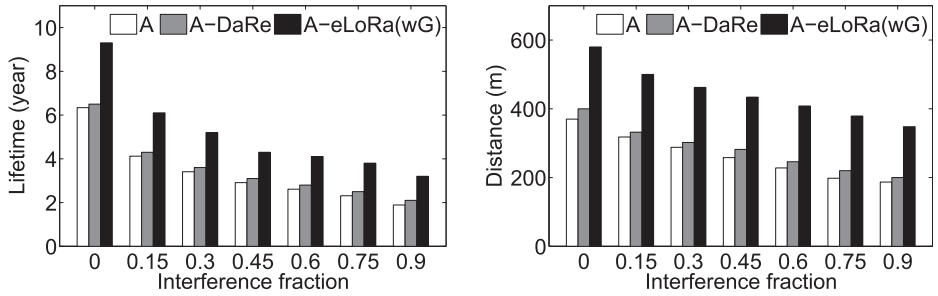
6.5 Detailed Performance

We evaluate the detailed performance improvement of eLoRa in two conditions: a static condition where the link quality keeps stable and a dynamic condition where the link quality is changed dynamically.

6.5.1 Static Condition. Figure 11 shows the impact of how the communication range and lifetime vary with the increasing number of gateways. We can see that the improvement of eLoRa keeps steady. With more gateways, eLoRa can achieve better performance than LoRaWAN and DaRe, e.g., 45.3% and 43.1% improvements on average over DaRe in terms of the communication range and lifetime when using three gateways. Note that by default LoRaWAN and DaRe do not utilize the combination of multiple gateways; therefore, their performance stays the same.

Figure 12 shows the impact of traffic load on the overall performance. Results show that eLoRa still achieves the best performance over LoRaWAN and DaRe with the increasing traffic load. Specifically, with a smaller traffic load, eLoRa achieves larger performance improvement, e.g., increasing the communication range and lifetime of DaRe by 45.6% and 50.3% when the traffic load is 80 bytes/h. We note that eLoRa can meet the communication range requirement when the traffic load is 80 bytes/h and achieve the longest distance and lifetime at a larger traffic load, while DaRe and LoRaWAN cannot meet the communication range requirement all the time, because with the use of rateless coding and cross-layer parameter optimizations, eLoRa can adapt the parameters to the network traffic load.

Table 4 shows the reliability, i.e., PRR, of default LoRaWAN, DaRe, and eLoRa at different communication distances. Compared with default LoRaWAN and DaRe, eLoRa always has the maximum



(a) Lifetime. (b) Communication range.

Fig. 13. Impact of dynamic interference.

Table 4. The PRR Comparison in Different Distances

Distance (m)	A-LoRaWAN	A-DaRe	A-eLoRa(wG)
600	25.3%	42.4%	58.1%
800	4.2%	19.6%	29.2%
1,000	0%	0%	14.8%
1,200	0%	0	0

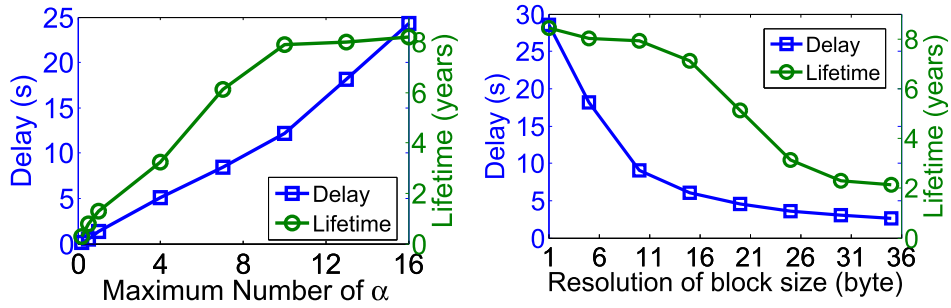
PRR. This is because eLoRa utilizes rateless codes to recover the lost packets under any patterns, e.g., bursty packet loss, which DaRe cannot handle.

From the table, we can see that when the distance reaches 1.2km, the PRRs of different mechanisms are all reduced to 0%. There are mainly two reasons for the relatively short communication distance in our evaluation. First, the hardware we used, i.e., the LoRa shield and LoRa gateway, is limited. The gain of the antenna of the hardware is only 3dBi [14], which seriously shortens the upper limit of the communication distance. This is proved by existing works that with a weak antenna, the order of magnitude of the distance between the end-device and the gateway at which packets started to get lost is 100 m. It is possible to achieve a kilometer-scale communication range if we implement eLoRa on the hardware with a high-gain antenna (e.g., 35 dBi using SMA Male Antenna [36]), which is indeed a further direction of our work. Second, the wireless link quality of our testbed is relatively poor. There are some unavoidable obstructions on the link, such as trees and buildings, which shortens the communication distance [10].

6.5.2 Dynamic Condition. Figure 13 depicts the overall performance under dynamic link qualities. We see that eLoRa can achieve the longest distance and lifetime over Class A and DaRe under different interference patterns. Specifically, eLoRa improves the communication range and lifetime of DaRe by 74% and 52.2% at the most frequently changing interference, because eLoRa can adapt the parameters to the link quality changes timely and therefore keeps the high performance, while for LoRaWAN and DaRe, their performance is drastically degraded when the interferer is active due to their fixed parameter settings.

6.6 Overhead of Optimization

Figure 14 presents the overhead of performing optimizations considering the parameter searching space. Figure 14(a) shows that the running time increases when the range of α is larger. When the range is bigger than 10, the increase of the optimized lifetime becomes small, which indicates that setting α to 10 can satisfy most optimization scenarios. The overhead is thus up to 12.13s. Figure 14(b) shows that with the coarse-grained resolution of x_n, the running time is small. We



(a) Impact of α. (b) Impact of block size x_n.

Fig. 14. Overhead of performing optimization.

Table 5. Impact of Optimizing the Lifetime

R>60%, L<5h								
M (m)	200	400	600	800	1,000	1,200	1,400	1,600
Opti_T	7.13	5.34	4.02	NA	NA	NA	NA	NA
M>600m, L<5h								
R (%)	20	30	40	50	60	70	80	90
Opti_T	8.5	8.02	7.76	5.89	4.23	2.13	NA	NA
M>600m, R>60%								
L (h)	3	5	7	9	11	13	15	17
Opti_T	NA	6.2	7.4	8.73	9.31	10.02	12.22	12.23

Table 6. Impact of Optimizing the Communication Range

R>60%, L<5h								
T (year)	2	5	8	11	14	17	20	23
Opti_M	822	613	423	198	NA	NA	NA	NA
T>6 years, L<5h								
R (%)	20	30	40	50	60	70	80	90
Opti_M	1,322	1,123	978	659	345	NA	NA	NA
T>6 years, R>60%								
L (h)	3	5	7	9	11	13	15	17
Opti_M	NA	234	432	578	765	912	1,078	1,298

note that when the resolution is lower than 10 bytes, the optimization amount of increased lifetime is small (about 5.1% increase). Therefore, in our evaluation, the resolution of x_n is selected to 10 bytes and the running overhead is up to 9.06s. The optimization is multiple orders of seconds, but considering that the common uplink transmission limitations of LoRaWAN applications are 30 seconds on-air time per day per device [21], the overhead is negligible. Besides, the optimization is only performed when the requirements are violated, which happens rarely.

6.7 Impact of Optimization Constraints

Tables 5 and 6 show the impact of constraint settings on the optimization results. Developers can choose proper constraint settings from the table when deploying eLoRa in practical scenarios. Although the concrete parameters may differ because of the utilized hardware, the trend is the same. Note that in this experiment the constraint is strictly fixed and there will be no results when

constraints cannot be satisfied. Table 5 shows the impact of varying constraints of communication distance (M), packet reception ratio (R), and transmission latency (L) on the optimization results of the lifetime (T). Table 6 presents the results of optimizing M while varying the other three constraints, T, R, and L. The results of the above two tables show that it is more likely to have the optimization solutions when the constraints are more relaxed. For example, when optimizing the lifetime T, setting M to be lower than 600m, R to be lower than 60%, and L to be bigger than 5 hours, it is highly possible to optimize T for 4 years at least.

7 CONCLUSION

In this article, we propose a practical system, eLoRa, for COTS devices. It utilizes rateless codes and jointly decoding with multiple gateways to extend the communication range and lifetime of LoRaWAN. To further improve the performance of LoRaWAN, eLoRa optimizes parameters from the PHY layer (e.g., spreading factor) and the link layer (e.g, block length).

eLoRa provides an attractive feature that network operators can define application-level requirements (e.g., lifetime and communication range). eLoRa periodically checks whether the requirements are violated and automatically carries out the parameter optimization based on the network model and the monitored network states.

We implement eLoRa on COTS LoRa devices and conduct extensive experiments on an outdoor testbed to evaluate the effectiveness of eLoRa. Results show that eLoRa can effectively improve the communication range of DaRe and LoRaWAN by 43.2% and 55.7% with a packet reception ratio higher than 60%, and increase the expected lifetime of DaRe and LoRaWAN by 18.3% and 46.6%. In the future, there are multiple directions to explore. First, we would like to optimize the degree distribution of LT code given the multiple gateway combination approach. Second, we would like to design a better gateway deployment strategy to cover more LoRa clients.

REFERENCES

[1] 2022. LoRaWAN 1.1 specification. Retrieved August 15, 2022 from https://lora-alliance.org/resource_hub/lorawan-specification-v1-1.

[2] 2022. NB-IoT. Retrieved August 15, 2022 from https://www.3gpp.org/news-events/1785-nb_iot_complete.

[3] 2022. Sigfox. Retrieved August 15, 2022 from https://www.sigfox.com.

[4] Adwait Dongare, Revathy Narayanan, Akshay Gadre, Anh Luong, Artur Balanuta, Swarun Kumar, Bob Iannucci, and Anthony Rowe. 2018. Charm: exploiting geographical diversity through coherent combining in low-power wide-area networks. In *Proceedings of the 17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN'18)*.

[5] Rashad Eletreby, Diana Zhang, Swarun Kumar, and Osman Yağan. 2017. Empowering low-power wide area networks in urban settings. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM'17)*.

[6] P. J. Marcelis, V. Rao, and R. V. Prasad. 2017. DaRe: data recovery through application layer coding for LoRaWAN. In *Proceedings of the Second International Conference on Internet-of-Things Design and Implementation (IoTDI'17)*.

[7] Gu Chaojie, Tan Rui, Lou Xin, and Niyato Dusit. 2018. One-hop out-of-band control planes for low-power multi-hop wireless networks. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM'18)*.

[8] M. Bor and U. Roedig. 2017. LoRa transmission parameter selection. In *Proceedings of the IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS'17)*.

[9] Thiemo Voigt, Martin Bor, Utz Roedig, and Juan Alonso. 2017. Mitigating Inter-network Interference in LoRa Networks. In *Proceedings of the 2017 International Conference on Embedded Wireless Systems and Networks (EWSN'17)*.

[10] Martin C. Bor, Utz Roedig, Thiemo Voigt, and Juan M. Alonso. 2016. Do LoRa low-power wide-area networks scale? In *Proc. of ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM'16)*.

[11] F. Van den Abeele, J. Haxhibeqiri, I. Moerman, and J. Hoebeke. 2017. Scalability analysis of large-scale LoRaWAN networks in ns-3. *IEEE Internet of Things Journal* 4, 6 (2017), 2186–2198.

[12] Juha Petäjäjärvi, Konstantin Mikhaylov, Marko Pettissalo, Janne Janhunen, and Jari Iinatti. 2017. Performance of a low-power wide-area network based on LoRa technology: Doppler robustness, scalability, and coverage. *International Journal of Distributed Sensor Networks* 13, 3 (2017), 1–16.

[13] A. Augustin, J. Yi, T. Clausen, and W. M. Townsley. 2016. A study of LoRa: Long range low power networks for the internet of things. *Sensors* 16, 9 (2016), 1–18.
[14] 2022. Dragino Lora Shield and Gateway. Retrieved August 15, 2022 from http://www.dragino.com/.
[15] Weifeng Gao, Zhiwei Zhao, and Geyong Min. 2020. AdapLoRa: resource adaptation for maximizing network lifetime in LoRa networks. In *Proceedings of the IEEE 28th International Conference on Network Protocols (ICNP'20)*.
[16] Di Mu, Yitian Chen, Junyang Shi, and Mo Sha. 2020. Runtime control of LoRa spreading factor for campus shuttle monitoring. In *Proceedings of the IEEE 28th International Conference on Network Protocols (ICNP'20)*.
[17] Sezana Fahmida, Venkata P. Modekurthy, Mahbubur Rahman, Abusayeed Saifullah, and Marco Brocanelli. 2020. Long-lived LoRa: prolonging the lifetime of a LoRa network. In *Proceedings of the IEEE 28th International Conference on Network Protocols (ICNP'20)*.
[18] Amalinda Gamage, Jansen Christian Liando, Chaojie Gu, Rui Tan, and Mo Li. 2020. LMAC: efficient carrier-sense multiple access for LoRa. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking (MobiCom'20)*.
[19] Chenning Li, Hanqing Guo, Shuai Tong, Xiao Zeng, Zhichao Cao, Mi Zhang, Qiben Yan, Li Xiao, Jiliang Wang, and Yunhao Liu. 2021. NELoRa: Towards ultra-low SNR LoRa communication with neural-enhanced demodulation. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems (SenSys'21)*.
[20] Michael Luby. 2002. LT codes. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS'02)*.
[21] 2022. LoRa technology is connecting our smart planet. Retrieved August 15, 2022 from https://www.semtech.com/lora/lora-applications.
[22] M. Rossi, N. Bui, G. Zanca, L. Stabellini, R. Crepaldi, and M. Zorzi. 2010. SYNAPSE++: Code dissemination in wireless sensor networks using fountain codes. *IEEE TMC* 9, 12 (2010), 1749–1765.
[23] W. Du, Z. Li, J. C. Liando, and M. Li. 2016. From rateless to distanceless: Enabling sparse sensor network deployment in large areas. *IEEE/ACM ToN* 24, 4 (2016), 2498–2511.
[24] Marco Zimmerling, Federico Ferrari, Luca Mottola, Thiemo Voigt, and Lothar Thiele. 2012. PTunes: runtime parameter adaptation for low-power MAC protocols. In *Proceedings of the 11th International Conference on Information Processing in Sensor Networks (IPSN'12)*.
[25] D. J. C. MacKay. 2005. Fountain codes. *IEE Proceedings - Communications* 152, 6 (2005), 1062–1068.
[26] Amin Shokrollahi. 2004. Raptor codes. In *Proceedings of the IEEE International Symposium on Information Theory, 2004 (ISIT'04)*.
[27] Semtech. 2015. AN1200.22: LoRa Modulation Basics.
[28] W. Dong, C. Chen, X. Liu, Y. He, Y. Liu, J. Bu, and X. Xu. 2014. Dynamic packet length control in wireless sensor networks. *IEEE Transactions on Wireless Communications* 13, 3 (2014), 1172–1181.
[29] 2022. RFM95. Retrieved August 15, 2022 from https://www.hoperf.com/modules/lora/RFM95P.html.
[30] W. Dong, J. Yu, and P. Zhang. 2015. Exploiting error estimating codes for packet length adaptation in low-power wireless networks. *IEEE Transactions on Mobile Computing* 14, 8 (2015), 1601–1614.
[31] G. Ferreira, B. Jesus, J. Vieira, and A. J. Pinho. 2011. Random block-angular matrices for distributed data storage. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'11)*.
[32] B. Reynders, W. Meert, and S. Pollin. 2016. Range and coexistence analysis of long range unlicensed communication. In *Proceedings of the International Conference on Telecommunications (ICT'16)*.
[33] 2022. SX1276/8 datasheet. Retrieved August 15, 2022 from https://www.datasheetq.com/SX1276-doc-Semtech/.
[34] 2022. ECLiPSe. Retrieved August 15, 2022 from http://eclipseclp.org/.
[35] 2022. USRP N210. Retrieved August 15, 2022 from https://www.ettus.com/all-products/un210-kit/.
[36] 2022. SMA Male. Retrieved August 15, 2022 from http://www.icstation.com/antenna-c-284_298.html.

Received 14 February 2022; revised 20 May 2022; accepted 5 June 2022