Deep Reinforcement Learning Based Coexistence Management in LPWAN

Md Ashikul Haque
Department of Computer Science
Wayne State University
USA

Abusayeed Saifullah
Department of Computer Science
Wayne State University
USA

Haibo Zhang
School of Computing
University of Otago
New Zealand

***Abstract*—This paper addresses the coexistence problem in Low-Power Wide-Area Networks (LPWANs), focusing on LoRa, a leading technology in this domain. Current LPWANs lack effective mechanisms to handle coexistence, especially in urban areas where numerous networks and devices may be operating in the limited spectrum. Existing approaches, including collision resolution techniques and adaptations from other wireless technologies like WiFi, are inadequate due to LPWANs' unique characteristics, such as long-range communication and severe energy constraints. Existing learning based approach addresses this issue through embedded Q-learning at low-power for LoRa nodes, imposing computation and energy overhead for them and limiting the network's learning capability by using simple Q-table learning at individual nodes. We propose a novel system design leveraging the computational capabilities of LoRa Network Servers (LNS) for coexistence management. By offloading learning and computation tasks to LNS, the proposed framework employs deep Q-learning, a powerful reinforcement learning technique, to adapt dynamically to complex coexistence scenarios. By exploiting the LNS's global view of the communication channels, our framework enables more effective learning and decision-making compared to decentralized approaches. Furthermore, our system design seamlessly integrates with traditional LoRaWAN infrastructure, imposing minimal overhead on low-power nodes. We evaluate our approach through physical experiments and large-scale simulations in NS-3, considering various coexistence scenarios for a LoRa network. Our results show that, in comparison with the state-of-the-art decentralized learning method, our scheme achieves up to 70.97%, 62.91%, and 47.01% of improvement in packet reception rate, energy per packet, and average transmission attempts per packet, respectively.**

***Index Terms*—deep learning, reinforcement learning, coexistence, experiment, simulation.**

I. Introduction

The *Low-Power Wide-Area Network (LPWAN)* technology is revolutionizing the Internet-of-Things (IoT) landscape by enabling low-power (milliwatts) wireless communication over long distances (kms). LPWANs are instrumental in enabling various wide-area IoT applications, such as smart cities, smart buildings, and smart metering, where numerous low-power nodes directly transmit data to a cloud-connected gateway [1, 2]. The escalating demand for IoT applications has led to the emergence of numerous LPWAN technologies in recent years, including LoRa (Long-Range), SigFox, IQRF, RPMA, DASH7, Weightless-N/P, Telensa in the ISM band, EC-GSM-IoT, NB-IoT in the licensed cellular band, and SNOW in the TV band [3].

The rapid proliferation of LPWANs in limited spectrum introduces coexistence challenges. With the number of connected devices expected to exceed 50 billions within a year [4], LPWANs are becoming ubiquitous, exacerbating coexistence issues, especially in urban areas. This results in degraded network performance, including reduced throughput, increased latency, and higher energy consumption, with some networks experiencing spectrum starvation. Research shows that throughput drops to 25% when four LoRa networks operate simultaneously [5], and studies report a 100% collision rate among 1000 nodes when LoRa, SigFox, or IQRF coexist [6, 7].

Current LPWANs are ill-equipped to address the impending challenge of coexistence. The nodes typically feature minimal computation power, low memory, and rely on limited energy supplied by small batteries. Due to severe energy constraints of the devices, LPWANs typically employ very simple and low-overhead MAC (media access control) protocols. For instance, LoRa's MAC protocol, LoRaWAN, is based on pure ALOHA, lacking collision avoidance mechanisms. Existing collision resolution approaches rely on physical layer interventions, necessitating changes to LoRa gateways, and can resolve collisions of several packets [8, 9, 10, 11, 12]. Coexistence handling strategies developed for WiFi, traditional short-range wireless sensor networks (WSN), and Bluetooth [13, 14] are inadequate for LPWANs. In environments with massive crowds of coexisting networks, detecting interference patterns can be challenging for LPWAN nodes, which may face an unprecedented number of hidden nodes due to their long communication ranges. Thus, conventional approaches such as TDMA (time division multiple access) or CSMA (carrier sense multiple access) are ineffective.

We explore and propose solutions to enable the coexistence of LPWANs in this paper, with a specific focus on LoRa, widely recognized as an LPWAN leader globally with over 600 use cases [15]. As LoRa operates in the unlicensed ISM band, shared by various other LPWAN technologies, coexistence management becomes imperative. Although some efforts, such as [16], propose addressing this issue through embedded Q-learning agents for LoRa, their approach introduces computa-

tion and energy overhead for LoRa nodes and limits the network's learning capability by using simple Q-table learning at individual nodes. Furthermore, their reward function depends on acknowledgments (ACK) from the gateway, introducing a serious weakness for the Q-learning agent, as the ACK packet can get lost. A centralized approach that entails the learning agent at the gateway can address this weakness. Moreover, their approach does not handle ***downlink*** (gateway to node) coexistence, which is important because lost ACKs cause retransmissions by the nodes, leading to depleted batteries.

We propose a learning framework for LoRa network coexistence, leveraging the computational capabilities of the LoRa Network Server (LNS). Given the dynamic and unknown wireless environment due to numerous unidentified networks, a learning-based approach is essential. However, the computational and memory demands of machine learning algorithms pose challenges for LoRa nodes. To address this, we offload learning and computation to the LNS. This offloading raises issues regarding performance measurement, parameter communication overhead, and time synchronization. Our centralized learning approach tackles these by introducing a MAC layer with novel techniques: piggybacking next transmission information with the ACK, managing ACK reception time slots at the node to minimize unnecessary Rx radio usage, and implementing a soft time synchronization method with minimal node overhead. The LNS, interacting with the communication channel through all nodes, facilitates better learning. We adopt deep Q-learning, a robust reinforcement learning (RL) technique, for handling coexistence in LoRa networks due to its ability to model complex scenarios. Our framework uses successful packet decoding at the gateway to assess the communication channel, enabling the RL agent at the LNS to select actions with a higher probability of successful transmissions.

While Q-learning-based approaches have been previously employed in frame-based MAC protocols with time synchronization to learn contention and collision within the same network [17, 18], we extend Q-learning to address coexistence with numerous unknown and uncoordinated LPWANs. Although a decentralized Q-learning scheme using simple Q-table has been proposed in [16], it limits the learning capability because each node learns based solely on its local knowledge rather than global knowledge. Our proposed learning framework integrates seamlessly with traditional LoRaWAN, requiring no modifications to the physical layer and imposing minimal overhead on low-power nodes. Specifically, this paper presents a deep-learning based co-existence handling technique for a LoRa network with other networks for both uplink and downlink by modeling coexistence management as a Markov Decision Process and using deep Q-learning in the LNS, requiring little to no energy overhead at the LoRa nodes. We evaluate our approach through both physical experiments and simulations in NS-3 [19], considering various coexistence scenarios for a LoRa network. Our results demonstrate that, compared with the state-of-the-art decentralized learning scheme, our scheme achieves up to 70.97%, 62.91%, and 47.01% improvements in packet reception rate, energy per node, and average transmission attempts per packet, respectively.

The rest of this paper is organized as follows: Section II reviews the related works. Section III firstly gives an overview of LoRa and then presents our system model. Section IV presents the detailed design of our deep learning framework and Section V outlines its implementation. In Section VI, we present the solutions to handle the key design challenges. Sections VII presents experimental and simulation results, and the paper is concluded in Section VIII.

II. Related Work

While coexistence handling has been studied for WiFi, WSN, and Bluetooth [13, 14], it is inadequate for LPWANs due to the latter's unique characteristics such as long-range communication and severe energy constraints. In environments characterized by massive crowds of coexisting networks, the interference pattern can be challenging to detect for LPWAN nodes. Some studies have explored resolving collided packets in LoRa through physical layer approaches [8, 9, 10, 11, 12, 20, 21, 22]. These approaches propose reactive solutions requiring alterations to the LoRa gateway and/or additional software-defined radio-based hardware. However, their effectiveness has primarily been demonstrated for collisions involving several packets, limiting their generalizability to handle the collisions from numerous unknown coexisting devices and networks. In contrast, we propose a link-layer approach for managing extensive coexistence, directly applicable to low-power LoRa nodes. The work proposed in [23] examines LoRa and WiFi coexistence in the 2.4GHz band but is not suitable for coexistence with networks other than WiFi. In an uncoordinated environment with collisions from various unknown networks, such an approach becomes ineffective and requires additional hardware.

Machine learning approaches have demonstrated notable effectiveness across various wireless applications including cognitive radios utilization [24], WSN routing [25], quality of service provisioning [26], and resource management [27, 28, 29, 30]. In the context of LPWAN, the efficacy of Q-learning has been investigated through simulation [31]. Additionally, Q-learning has been incorporated into frame-based MAC protocols featuring time synchronization. This implementation allows Q-learning to adeptly learn contention and collision patterns among nodes within the same network utilizing a single channel [17, 18].

Very few recent works have addressed coexistence and jamming mitigation in LPWAN [16, 32, 33]. The work in [16] presents a Q-learning approach for LoRa nodes to coexist with other ISM band users. However, they use a Q-table with a simple reward function to find suitable transmission time, channel, and spreading factor, which is not effective for complex environments. Furthermore, it is implemented in LoRa node depleting its energy for calculation related to Q-table. The study in [32] presents jamming mitigation through non-cooperative reinforcement learning and game theoretic

interaction between the jammer and the LPWAN network. However, there is no such advisory in the coexistence scenario making this approach unsuitable. In contrast, we propose to use a deep Q-Network (DQN) approach with complex reward function comprising transmission power, coding rate, and packet reception status for uplink transmission (i.e., LoRa node to gateway). This approach can handle complex coexistence environment while enabling lower energy consumption. Moreover, it is implemented centrally at the network manager waiving the computation load from LoRa nodes. Furthermore, we propose to use a deep Q-network for coexistence in downlink communication (i.e., acknowledgment from gateway to LoRa node), as reliable downlink communication can help the node receive its acknowledgment promptly and alleviate the need for unnecessary retransmissions. Our approach is designed to assist LoRa nodes to handle coexistence with numerous unknown and uncoordinated LPWANs with little energy overhead.

III. Background and System Model

Here, we describe the necessary background for LoRa in Section III-A and the system model in Section III-B.

A. An Overview of LoRa

LoRa is a prominent commercially available LPWAN technology that enables long range (3-7 miles) [34], low power communication for energy-constrained IoT devices. Its distinguishing feature lies in its ability to efficiently receive packets even amidst low signal-to-noise ratios (SNR). This capability is achieved through Chirp Spread Spectrum (CSS) modulation, a technique that disperses the signal across the entire bandwidth spectrum, thus fortifying it against interference and enabling reception under challenging SNR conditions. The modulated signal comprises symbols or chirps, characterized by a continuous frequency variation over time. Each chirp carries encoded information through multiple cyclic shifted chips. The spreading factor (SF), a crucial parameter ranging from 7 to 12, governs the number of chips per symbol, thereby influencing the data transmission rate, airtime, and energy consumption. Notably, higher SF translates to lower data rates and increased energy consumption, while lower SF results in higher data rates and reduced energy consumption.

LoRa's adaptability extends to channel allocation, bandwidth, and coding rate configurations. Operating within the unlicensed ISM band (902-928MHz) in the United States, LoRa defines 64 uplink channels with 125kHz bandwidth, alongside an additional 8 uplink channels featuring 500kHz bandwidth. Downlink operations (gateway to nodes) are facilitated through 8 channels, each possessing a bandwidth of 500kHz. Furthermore, LoRa offers various levels of forward error correction (FEC), represented by coding rates ranging from $\frac{4}{5}$ to $\frac{4}{8}$, with higher coding rates ensuring enhanced reliability at the expense of prolonged packet duration.

LoRaWAN (LoRa Wide Area Network) is the MAC protocol of LoRa, facilitating low-power, low data rate communication among numerous end-devices or nodes. These nodes form a star topology by directly connecting to one or more gateways, which serve as intermediaries for relaying data to a central LoRa Network Server (LNS). The LNS is responsible for managing network parameters, ensuring security, and addressing application requirements. On the other hand, the application server interprets sensor data and application information provided by the sensors.

B. System Model

We consider a landscape of dense deployment where a LoRa network, so-called a *primary network* , coexists with many other networks, so-called *secondary networks*, in the same spectrum. The secondary networks can be based on LoRa or any other technologies. We aim to optimize the resource allocation for the primary network to mitigate the interference from the secondary networks.

LoRa follows a star topology, with nodes transmitting their packets directly to gateways. The nodes are low-powered in terms of both computation and energy. Typically, these nodes sleep until required to transmit packets to the gateway. Node transmissions rely on acknowledgments (ACK) from the gateway for confirmation. If the nodes do not receive any ACK within the ACK period, they initiate packet retransmission with a configurable limit on the number of retries. Additionally, nodes intermittently assess their surroundings, autonomously determining whether data transmission is warranted, thus enforcing a minimum inter-arrival time between packets. It's worth noting that the primary flow of traffic within an LPWAN is from nodes to the gateway (uplink direction), and nodes have the ability to transmit across multiple channels as dictated by the link-layer protocol. The gateway is line-powered and Internet-connected. It supports concurrent reception of multiple packets on the same channel using different SFs, with the maximum number of simultaneous receptions governed by hardware specifications.

Nodes in the secondary networks face similar constraints to those in the primary network, including energy limitations and inability to sustain continuous transmission across all channels. The multitude of secondary networks within the spectrum significantly impacts the primary network's performance. While these secondary networks may employ different communication technologies, they all operate within the same spectrum. It's crucial to differentiate between coexistence and jamming; while jamming involves intentional interference causing collisions, coexistence stems from the finite spectrum leading to unintended collisions.

IV. Leveraging Deep Learning to Handle Coexistence

This section outlines the overview of our approach and models both uplink and downlink coexistence as Markov Decision Process (MDP).

A. Overview

In the LPWAN coexistence scenario, competing networks are uncoordinated, meaning transmission success depends on

current environmental conditions. With numerous secondary networks and diverse applications, the environment is largely unknown to primary network nodes, making a learning-based approach effective.

Unlike other wireless technologies, LoRa achieves orthogonality with different Spreading Factors (SFs). Choosing best-effort combinations of channel, SF, timing, and transmission power can improve transmission success. However, the large number of combinations makes learning challenging, especially with many secondary networks. Selecting higher SF and Tx power can impact battery life. Deep reinforcement learning (RL) can significantly improve the selection of suitable communication parameters while minimizing the nodes' energy consumption.

Deep Q-learning, though effective, requires high computational power and energy, impractical for LoRa nodes. Offloading computation to the LoRa Network Server (LNS) addresses this but introduces challenges in measuring performance, communicating parameters, and synchronizing time. Our design addresses the challenges associated with a centralized learning approach.

We propose a centralized learning agent based on RL to enhance LoRa network performance amid numerous independent networks. Centralized learning provides global environmental knowledge, speeding up the learning process. The network manager, aware of parameter assignments, can predict low-collision parameters based on past transmission, aiding action selection. Utilizing deep Q-learning, our agent learns best-effort actions through trials, quantified by Q-values. Our goal is to enhance communication quality and conserve LoRa node energy by optimizing transmission timing, SF, and Tx power for uplink, improving downlink quality to avoid retransmissions, and reducing computation overhead.

The Centralized Q-learning agent at the LNS selects best-effort communication paths (channel, SF, timing, Tx power) based on observations and learning. The RL agent interacts with the environment, receiving rewards or penalties based on the successful reception of LoRa packets, encouraging efficient Tx power and SF use. Through trial evaluations, the agent trains the deep Q-network, significantly enhancing performance in coexistence scenarios.

B. Deep Reinforcement Learning

We propose using reinforcement learning (RL) to find high-performing uplink and downlink communication parameters for LoRa networks in coexisting scenarios. RL is a machine learning technique that enables an agent to learn from its own experience and adapt to the environment. The agent interacts with the environment by taking actions and receiving rewards or penalties based on the outcomes. The agent aims to maximize cumulative rewards by learning the best-effort policy for mapping environmental states to actions.

Downlink communication (gateway to nodes) has been overlooked for coexistence scenario in previous works. Even though this is justified in a sense that the gateway is not energy-constraint, it has energy impact on the LoRa node when an ACK gets lost and the LoRa node retransmits the already successfully transmitted packet. This energy overhead for LoRa nodes can be avoided by introducing coexistence in downlink communication.

We consider the LNS as our RL agent, which collects data from all the gateways in the network and assigns communication parameters to the LoRa devices for uplink and to the gateways for downlink. The LNS observes the state of the network, including packet reception for each LoRa device. Determining the quality of a downlink channel from gateway side is simple as the gateway can observe the retransmissions from the nodes. If a successfully transmitted packet gets retransmitted, it implies the ACK got lost. From this observation it can determine the last downlink channel parameters and transmission timing were being used by some other coexisting network.

The LNS then takes an action, which involves changing the communication parameters of LoRa devices, and receives a reward or penalty based on the outcome. For uplink, the reward is positive for successful packet reception but negative for higher Tx power or higher SF. The LNS learns a Q-function, which estimates the expected future reward for each state-action pair, and follows an epsilon-greedy exploration strategy to balance between exploiting the best-known actions and exploring new actions. The LNS sends communication parameters (transmission time, channel, SF, Tx power) to the LoRa device via the downlink channel with the ACK.

In this scenario, a centralized agent, embodied by the LNS, oversees the communication paths for both uplink and downlink within the LoRa network. The MDP is characterized by a 4-tuple (S, A, T, R), where S, A, T, and R are the set of states, set of actions, state-transition function, and the set of rewards, respectively.

1) State Set: The state of the network manager encompasses the parameters of the communication medium for the entire network, considering characteristics such as channels and SFs, as well as the aggregated state of the generated packets across all nodes. In the LoRa network, we use $s_i =< c_i, sf_i, t_i, tx_i >$ to represent the state for node or gateway i, where $c_i \in C_{\text{up}}$ (node) or $c_i \in C_{\text{down}}$ (gateway), $sf_i \in SP$, $t_i \in T$, $tx_i \in TXP$ and C_{up}, C_{down}, SP, T, and TXP represent the sets of uplink channels, downlink channels, SFs, transmission timings, and Tx powers respectively. The state of the uplink and downlink network can be represented by $S_u = \{s_1, s_2, \ldots, s_n\}$ and $S_d = \{s_1, s_2, \ldots, s_g\}$, respectively, n being the number of nodes and g being the number of gateways in the network. As Tx power in ISM band is limited (30 dBm in the USA) by the respective authority in each country, the set of Tx powers is same for both uplink and downlink.

2) Action Set: The LNS's actions involve generating uplink and downlink communication paths for all nodes and gateway, respectively. Actions include selecting channels $c_i \in C_{\text{up}}$ or $c_i \in C_{\text{down}}$, spreading factors $sf_i \in SP$ and Tx power $tx_i \in TXP$ for transmission, and selection of transmission timing $t_i \in T$. The action for node or gateway i in the network is

defined as $a_i = \{c_i, sf_i, t_i, tx_i\}$. The set of actions for nodes and gatewayas in the primary network is represented by $A_u = \{a_1, a_2, \ldots, a_n\}$ and $A_d = \{a_1, a_2, \ldots, a_g\}$, respectively, n being the number of nodes and g being the number of gateways in the network.

3) State Transition Function: The state transition function F_u and F_d models the dynamics of the environment, defining the next state for each state-action pair with a probability. In this case, $F_u : S_u \times A_u \longrightarrow \mathbb{P}(S_u)$ and $F_d : S_d \times A_d \longrightarrow \mathbb{P}(S_d)$, where $\mathbb{P}(S_u)$ and $\mathbb{P}(S_d)$ are probability distribution over the set of states in uplink and downlink, respectively. The transition probability $\mathbb{P}[s_u{}'|s_u, a_u]$ and $\mathbb{P}[s_d{}'|s_d, a_d]$ adheres to the Markov Property, signifying that the next state depends solely on the immediate preceding state and action.

4) Reward Function Formulation: Uplink communication focuses on better packet delivery while being energy efficient as much as possible. To achieve that we need to select the combination of communication parameters (i.e., channel, SF, Tx power, and transmission timing) incurring low energy (lower SF and Tx power) and having higher probability of successful packet transmission. Moreover, the coding rate used for communication can impact energy consumption due to variable packet size incurring extra computation at the nodes. Thus, we need to give higher reward whenever the packet is delivered successfully using lower SF and Tx power, and vice versa.

For every action a_u taken in state s_u, the agent receives an immediate reward or penalty r_u. The reward function is defined as follows:

$$r_u(s_u, a_u) = \begin{cases} k_1 \left(\frac{1}{p \times sf}\right) + k_2 \cdot cd & \text{packet received} \\ -k_3 \left(p \times sf\right) - k_4 \cdot cd & \text{packet not received} \end{cases}$$

Here, k_1, k_2, k_3, k_4 are coefficients to normalize the values, p is the Tx power, sf is the SF, and cd is the coding rate. The reward function incorporates a penalty for higher Tx power and SF even when the packet is received successfully. This encourages to seek a communication path with lower Tx power and SF, optimizing the trade-off between successful transmission and resource utilization. The agent's objective is to find a policy π_u^* maximizing its total reward.

Downlink communication does not focus on conserving energy rather it tries to increase the probability of successful packet transmission. Thus, the downlink communication focuses on choosing communication parameters with higher probability of successful packet delivery and gives reward whenever a packet gets delivered irrespective of the Tx power and SF used for it. Our proposed reward function is as below:

$$r_d(s_d, a_d) = \begin{cases} 1, \text{ ACK is successfully delivered} \\ -1, \text{ ACK is not successfully delivered} \end{cases}$$

The objective is to find a policy π_d^* maximizing its total reward.

5) Deep Q-Network Design: The uplink and downlink RL modeled above can integrate Deep Q-Network (DQN), involving a neural network architecture that enables the LNS to learn and optimize communication parameters. The Deep Q-Learning network is designed to map state-action pairs to their corresponding Q-values.

The network design for Deep Q-Learning in the uplink and downlink RL Module includes an input layer for uplink and downlink, receiving node ID, channel, SF, coding rate, Tx power, packet reception status, transmission timing, and gateway ID, channel, SF, packet reception status, transmission timing, respectively. There are four fully connected hidden layers with Rectified Linear Unit (ReLU) activation functions, although the number of hidden layers and neurons per layer can be adjusted based on network complexity.

The output layer provides Q-values for each possible action: next channel, next SF, next transmission timing, next Tx power, next coding rate, and node ID (uplink), or next channel, next SF, next transmission timing, and gateway ID (downlink). Only Q-values for the current node (uplink) or gateway (downlink) ID are used during training and decision-making, ensuring optimization for the specific node or gateway.

6) Learning Algorithm: The LNS learns a Q-function to estimate the expected future reward for each state-action pair. The LNS can follow an epsilon-greedy exploration strategy to balance between exploiting the best-known actions and exploring new actions.

V. Design of the Q-Learning System

This section presents the integration of our proposed DQN in the LoRa network. The proposed design integrates reinforcement learning (RL) modules into the existing LoRa network architecture to optimize both uplink and downlink communication parameters. The system consists of three modules at the LNS (*Uplink RL Module*, *Downlink RL Module,* and *Downlink RL Transmission Module*) and one module at the LoRa Nodes (*Downlink RL Reception Module*).

A. Uplink RL Module

The Uplink RL Module operates at the LNS, acting as the RL agent overseeing all the LoRa nodes. This module is responsible for optimizing the uplink communication parameters for individual nodes based on the information received from the decoder and the current state of the network. This module works as follows:

For each packet reception, this module collects information from the LoRa decoder, including whether the packet was successfully decoded or not, as well as the node ID and communication parameters such as channel, SF, and Tx power. Additionally, the current system time is obtained and the transmission time is then calculated by estimating the approximate packet air time, packet reception and decoding time.

This information is then fed into the DQN designed in IV-B, which will output the next action (i.e., channel, SF, transmission period, Tx power) for the node from which the packet was received. If the parameters in the suggested action is different from the ones the node is currently using, the suggested action is then passed to the Downlink RL Transmission Module for further processing.

B. Downlink RL Module

The Downlink RL Module operates at the LNS, acting as the RL agent overseeing all the downlink communications of the gateways. This module is responsible for optimizing downlink communication parameters for all the gateways based on the information received from the decoder and the current state of the downlink network.

The module receives information from the downlink RL transmission module along with all the details (i.e., channel, SF, transmission time, success or failure, gateway ID). This information is then fed into the DQN designed in IV-B, which suggests the next action (i.e., channel, SF, transmission time, gateway ID) for a specific gateway. The suggested action is then passed to the Downlink RL Transmission Module.

C. Downlink RL Transmission Module

The Downlink RL Transmission Module is responsible for the dissemination of the uplink communication parameters received from the Uplink RL Module. It operates at the LNS and aims to transmit the uplink communication parameters for each node.

After receiving the uplink action parameters that includes the uplink channel, SF, transmission timing, and node ID, the module appends this information at the end of the ACK packet. Note that the ACK packet is transmitted irrespective of the success or failure of the LoRa uplink communication. In the event of a failed uplink scenario, the ack bit is set to 0, while it is set to 1 for a successful uplink packet transmission by the node. The ACK packet is sent using the downlink communication parameters obtained from the downlink RL agent, which include the channel, SF, and non-periodic transmission timing.

The module then waits for any retransmission from the node. Note that the retransmission window is larger compared to the usual LoRaWAN to accommodate all the processing at LNS and to find a proper transmission time for downlink. If there is a retransmission, two cases are considered:

1) If the ACK was for a correctly received packet, it is considered a failed downlink communication (i.e., the packet is received, but the ACK is lost). The module informs the RL agent of the failure, providing communication parameters such as gateway ID, channel, SF, and transmission time.
2) If the ACK was for an incorrectly received packet, two sub-cases are considered: a) The packet is received at the given transmission parameters in the ACK. This is considered a successful downlink communication, and the module informs the RL agent of the success along with the communication parameters; b) The packet is not received at the given transmission parameters in the ACK. This is considered a failed downlink communication, and the module informs the RL agent of the failure along with the communication parameters.

The Downlink RL Transmission Module plays a crucial role in providing feedback to the RL agent for optimizing downlink communication strategies.

D. Downlink RL Reception Module

The Downlink RL Reception Module is situated at the LoRa node's MAC layer and complements the existing LoRaWAN protocol. This module reads the downlink ACK packet to extract the transmission parameters and the ACK bit.

Based on the ACK bit, the module adjusts the transmission parameters. If retransmission is needed (ACK bit is 1), the packet is sent again with the updated parameters. This allows the LoRa node to adapt to the RL agent's suggested communication parameters, optimizing downlink communication.

The combination of the Uplink RL Module, Downlink RL Transmission Module, and Downlink RL Reception Module forms a closed-loop system that continuously adapts and optimizes both uplink and downlink communication parameters in a coexisting LoRa network. This integration allows for efficient resource utilization and energy conservation in LoRa-based IoT networks.

VI. Handling the Key Design Challenges

In this section, we discuss the challenges of our proposed system. Centralized learning faces issues as communication parameters are sent to nodes via ACK packets. While the LNS can understand the overall network communication, it struggles to relay this to nodes due to potential clock differences, causing timing issues.

Managing downlink coexistence adds another layer of complexity. The gateway may delay ACK packet transmission, leading to scenarios where ACKs are sent after a delay. LoRa devices typically listen for ACKs during two short receive windows post-transmission. Since the Tx/Rx radio consumes significant energy, nodes must use their radios efficiently and cannot keep them on indefinitely.

A. Time Synchronization

Our learning framework requires time synchronization between the gateway and the LoRa nodes, but precise time synchronization is not needed. For our scenario, we opt for weak time synchronization, meaning the LoRa nodes only update their clock after receiving their ACK packet. Furthermore, we propose to estimate the LoRa device's time deviation from the LNS's perspective. To achieve this, we estimate the airtime of the LoRa packet, which can be calculated using SF (Spreading Factor), bandwidth, and packet size (parameters already known). Estimating the airtime alone is sufficient because the propagation time is usually less than 100 μs (corresponding to approximately 30 km) for most LoRa applications.

Note that the transmission timing sent through the ACK already accounts for the ACK packet airtime (t_{ack}), meaning the LNS has already subtracted the ACK airtime from the transmission timing. This ensures that the node does not need to perform any computation; it can simply run the transmission timer according to its clock and transmit its packet when the timer reaches zero.

Suppose our calculated airtime is t_a and the time sent through ACK is t_d. Therefore, the estimated time at the LNS should be $t_{\text{est}} = t_{\text{ack}} + t_d + t_a$. Now, if the time at the LNS is

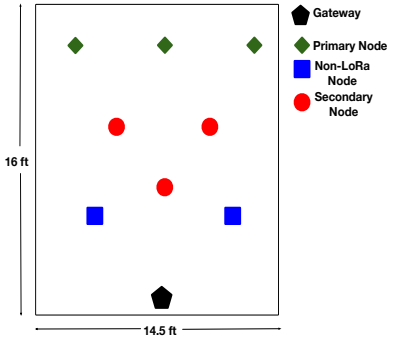


Fig. 1: Experimental setup.

t_{cur} and the difference between t_{cur} and t_{est} is zero, the clocks of the LNS and the node are synchronized. However, if there is an offset t_{off} between t_{cur} and t_{est}, this offset should be accounted for in future ACKs.

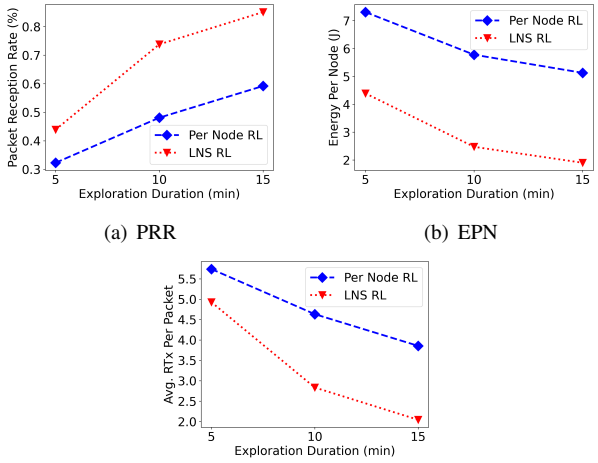
We propose using a simple regression model on the calculated time offset for each primary LoRa node at the LNS to account for clock drift in the nodes. This model is trained for each uplink packet, improving its estimation as the network continues to operate. While this method does not achieve perfect time synchronization, it gets very close to it. This approach offloads all responsibility to the LNS and ensures there is almost no overhead for the LoRa nodes.

B. ACK Reception at the Node

Managing downlink coexistence means that the gateway will not transmit the ACK packet immediately, which presents a new challenge. With downlink coexistence management, there can be scenarios where the ACK is transmitted after a delay following the output from the DQN. Since the time between the LoRa node and LNS is closely synchronized, we can leverage this synchronization to convey additional information about downlink ACK timing. When the LNS receives uplink communication parameters from the uplink DQN for the node, it adds them to the ACK along with estimated downlink communication parameters received from the downlink DQN.

LoRa devices usually collect sensor data and send the data periodically. Suppose the LoRa node produces a packet every t_{prd} time period (not transmission). Thus, the uplink communication transmission timing t_d should satisfy $t_{\text{prd}} < t_d < 2t_{\text{prd}}$, meaning the uplink DQN needs to provide uplink transmission for the node after the packet is ready to transmit and before the next packet is ready to transmit. Therefore, the estimated downlink transmission timing t_{down} chosen by the LNS should satisfy $t_d + t_a < t_{\text{down}} < 2t_{\text{prd}}$.

To address this challenge, the estimated downlink transmission timing t_{down} and other communication parameters are also included with the ACK. The node uses these communication parameters and timing to listen for the ACK of its next uplink transmission. Since the node and LNS are not perfectly time synchronized, the LNS transmits the ACK multiple times (before and after the downlink transmission timing). This approach ensures that, even if the node turns on the Rx radio a bit early or later, it can receive the ACK.



(a) PRR (b) EPN

(c) ATP

Fig. 2: Experimental results by varying exploration duration.

VII. Evaluation

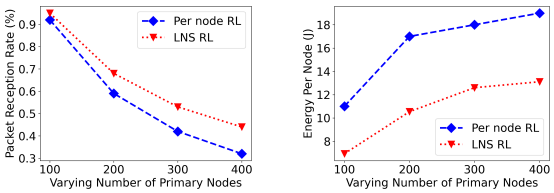
In this section, we evaluate the performance of our proposed system through both small-scale indoor experiments and large-scale simulations.

Setup: We implemented the deep Q-learning agent using Pytorch [35] following our approach. The number of neurons in the four hidden layers was set to 512, 256, 128, and 64, respectively. We used $\epsilon = 0.1$ and learning rate $\gamma = 0.5$.

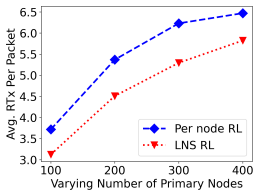
For simulations, we used the LoRaWAN NS3 module from [36] as the environment and connected it to our learning agent via NS3-AI [37]. We varied the number of primary and coexisting nodes, with up to 500 coexisting and 400 primary nodes, randomly placed within a 6 km radius disc. Performance comparisons of the Q-learning agent were made after the random exploration interval. Each exploration or training duration was followed by a five-hour network evaluation. All nodes and the gateway utilized 8 channels in the US 915MHz band, with coexisting nodes employing 8 retransmissions per packet to create severe coexistence. All simulations involved a single gateway.

For experiments, we used three USRP B200s [38] as one LoRa gateway and two non-LoRa coexisting nodes through GNU radio [39]. We employed six Dragino LoRa shields [40] with Arduino Uno R3 [41], where three were used as primary nodes and the other three as coexisting LoRa nodes. We use non-LoRa coexisting nodes to create more challenging scenario for the LNS, because all SFs gets interfered by the transmission of non-LoRa coexisting nodes. An M1 Macbook pro was used as the LNS. All the nodes and the gateway can only use one channel in the US 915MHz band to emulate severe density of traffic by large number of nodes. The nodes can use Tx power from 0-20 dBm and SF from 7-10. All the nodes transmit 100 packets per hour with up to eight retransmission per packet. The node deployment in our experiments is illustrated in Figure 1.

Metrics: we employ three metrics, namely packet reception rate (PRR), average energy per node (EPN), and average transmission attempts per packet (ATP).

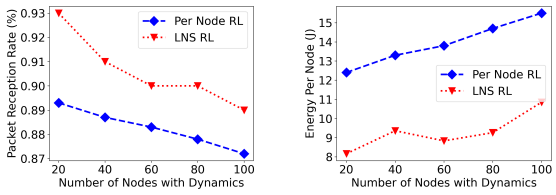


(a) PRR (b) EPN

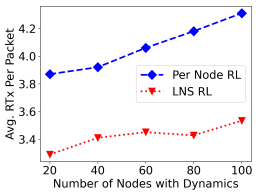


(c) ATP

Fig. 3: Results under varying number of primary nodes.



(a) PRR (b) EPN



(c) ATP

Fig. 4: Results under varying number of dynamic nodes.

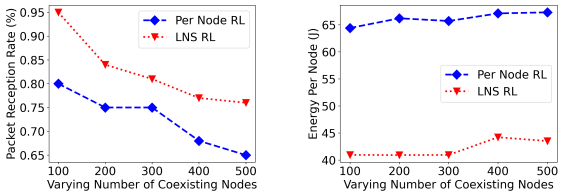
Baseline: We compare our scheme (marked as 'LNS RL') with the state-of-the-art work that handles coexistence in LoRa [16] (marked as 'Per Node RL'). This work [16] evaluated their approach compared to LoRaWAN and LoRaWAN performed poorly. Therefore, we are not comparing our approach to LoRaWAN in this paper.

A. Experimental Results

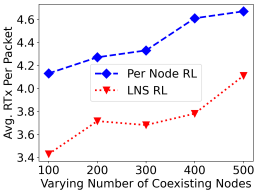
We perform a small-scale experiment using an indoor setup to evaluate our approach. We vary the exploration duration from 5 to 15 minutes for our experiment. Exploration duration refers to the lower epsilon value (0.1) which means there is a 90% probability of randomly selecting an action instead of using trained model to select the action. After exploration duration we evaluate our network for one hour. As depicted in Figure 2, our approach outperforms the 'Per Node RL' scheme by a large margin on PRR, EPN, and ATP. It is evident that our approach can explore and learn about the communication environment more quickly than Per Node RL. It occurs due to our centralized learning approach along with the use of DQN. Significant improvement in EPN is contributed by no computation overhead for LoRa nodes and low number of retransmissions due to downlink coexistence management.

B. Simulation Results

1) Performance Under Varying Number of Primary Nodes: In this simulation, we evaluate our approach by varying the number of primary nodes from 100 to 400. For each run, the number of coexisting nodes was equal to the number of primary nodes. Under this setup, we present the results in Figure 3. It can be seen that LNS RL achieves an improvement of 37.5% in terms of PRR. Additionally, Figure 3(b) demonstrates an improvement of around 33.72% on EPN when using LNS RL. Figure 3(c) shows that LNS RL achieves an improvement of around 10% in terms of ATP. The performance improvements in all three evaluation metrics increase slightly with the number of primary nodes, demonstrating the superior scalability of our approach.



(a) PRR (b) EPN

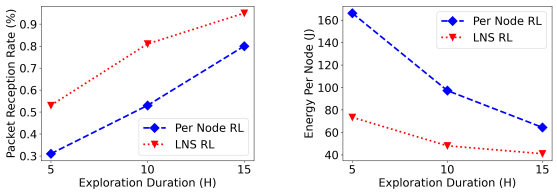


(c) ATP

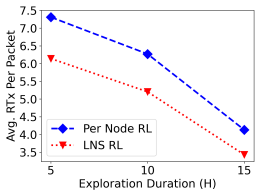
Fig. 5: Results under varying number of coexisting nodes.

2) Performance Under Varying Number of Coexisting Nodes: In this simulation, we utilized 100 primary nodes and varied the number of coexisting nodes from 100 to 500. As depicted in Figure 5(a), LNS RL achieves an improvement of 16.92% in PRR compared to Per Node RL when the number of coexisting nodes is 500. Figure 5(b) demonstrates around a 35.33% improvement in EPN with LNS RL maintaining an EPN of 43.52 J compared to 67.3 J in Per Node RL when the number of coexisting nodes is 500. Moreover, in Figure 5(c), LNS RL achieves a 12% improvement in ATP, maintaining a value of 4.11 compared to 4.67 in Per Node RL when the number of coexisting nodes is 500. These findings highlight the superior coexistence management capabilities of LNS RL, particularly under complex conditions, thanks to its utilization of deep learning for accurate scenario representation.

3) Performance under Dynamic Coexistence Traffic: In this simulation, we evaluated our approach under dynamic system conditions with a fixed number of 100 primary and coexisting nodes. We varied the number of dynamic coexisting nodes from 20 to 100. As depicted in Figure 4(a), LNS RL achieves a 2.06% improvement in PRR compared to Per Node RL when the number of dynamic coexisting nodes is 100. Figure 4(b) demonstrates around a 30% improvement in EPN with

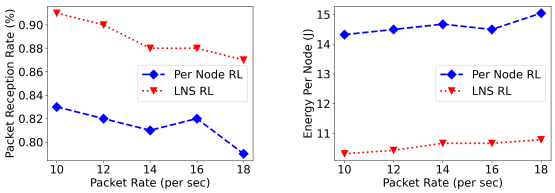


(a) PRR (b) EPN

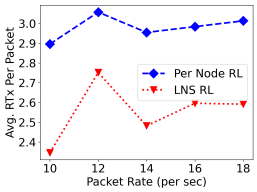


(c) ATP

Fig. 6: Results under varying exploration duration.



(a) PRR (b) EPN



(c) ATP

Fig. 7: Results under varying coexistence node traffic.

LNS RL maintaining an EPN of 10.85 J compared to 15.5 J in Per Node RL when the number of dynamic coexisting nodes is 100. Moreover, in Figure 4(c), LNS RL achieves an 18% improvement in ATP, maintaining a value of 3.53 compared to 4.31 in Per Node RL with 100 dynamic coexisting nodes. These results highlight the robustness and adaptability of our technique under dynamic scenarios, ensuring effective performance without overfitting.

4) Performance under Varying Exploration Duration: In this simulation, we evaluated our approach under varying exploration durations with a fixed number of 100 primary and coexisting nodes. As shown in Figure 6(a), LNS RL achieves a significant improvement of 70.97% in PRR compared to Per Node RL when the exploration duration is 5 hours. Figure 6(b) demonstrates a notable 55.82% improvement in EPN with LNS RL maintaining an EPN of 73.42 J compared to 166.19 J in Per Node RL with a 5-hour exploration duration. Moreover, in Figure 6(c), LNS RL achieves a 16% improvement in ATP, maintaining a value of 6.14 compared to 7.31 in Per Node RL with a 5-hour exploration duration. These findings highlight our approach's capability to quickly model complex communication paths, with LNS RL showing superior performance under shorter exploration duration.

5) Performance under Varying Coexistence Node Traffic: In this simulation, we evaluated our approach under varying node traffic by adjusting the total number of generated packets per second across primary and coexisting nodes. We fixed the number of primary nodes at 10 and coexisting nodes at 50, with specified random packet inter-arrival times. As shown in Figure 7(a), LNS RL achieves a 10.13% improvement in PRR compared to Per Node RL when the generated packet rate is 18 packets per second.

Figure 7(b) demonstrates a 28.31% improvement in EPN with LNS RL maintaining an EPN of 10.78 J compared to 15.04 J in Per Node RL at a packet rate of 18 packets per second. Moreover, in Figure 7(c), LNS RL achieves a 14% improvement in ATP, maintaining a value of 2.59 compared to 3.01 in Per Node RL at a packet rate of 18 packets per second. These results underscore the robustness of our approach in handling higher traffic scenarios and effectively capturing complex communication dynamics.

C. Discussion

From the above experimental and simulation results on varying scenarios, it is evident that our technique does not impose any energy overhead; rather, it saves energy compared to the state-of-the-art approach [16]. This is attributed to: 1) modifications in the LoRa MAC layer to facilitate necessary communication with minimal reception at the LoRa node, 2) utilization of weak time synchronization, 3) usage of deep learning at the LNS to predict other networks' transmissions, and 4) improving downlink transmission to help avoid unnecessary uplink transmissions due to lost ACKs.

VIII. Conclusion

In this paper, we have addressed the challenge of coexistence in LPWANs, focusing on developing a deep RL approach that can adapt dynamically to complex coexistence scenarios to improve communication quality for both uplink and downlink while saving energy at the LoRa node. Our RL framework leverages the computational capabilities of LoRa Network Servers (LNS) and its global view of the communication channels to effectively deal with coexistence. By offloading the learning and computation tasks to LNS, it seamlessly integrates with the traditional LoRaWAN infrastructure, imposing minimal overhead on low-power nodes. We have evaluated our approach through both physical experiments and large-scale simulations in NS-3, considering various coexistence scenarios for a LoRa network. Our results demonstrate that, compared with the state-of-the-art decentralized learning scheme, our scheme achieves up to 70.97%, 62.91%, and 47.01% of improvement in packet reception rate, energy per node, and average transmission attempts per packet, respectively. In the future, we will extend our learning framework for coexistence handling to other LPWANs beyond LoRa.

Acknowledgement

The work was supported by the US National Science Foundation through grants CNS-2301757, CAREER- 2306486, CNS-2306745, and by the US Office of Naval Research through grant N00014-23-1-2151.

References

[1] S. Fahmida and et al., "Long-lived lora: Prolonging the lifetime of a lora network," in *2020 ICNP*. IEEE, 2020, pp. 1–12.

[2] ——, "Real-time communication over lora networks," in *2022 IoTDI*. IEEE, 2022, pp. 14–27.

[3] A. Saifullah and et al., "Low-power wide-area networks over white spaces," *ToN*, vol. 26, no. 4, pp. 1893–1906, 2018.

[4] K. Mekkia and et al., "A comparative study of lpwan technologies for large-scale iot deployment," in *ICT Express*, 2018.

[5] T. Voigt and et al., "Mitigating inter-network interference in lora networks," ser. EWSN '17, 2017, pp. 323–328.

[6] L. Krupka and et al., "The issue of lpwan technology coexistence in IoT environment," in *ME*, 2016, pp. 1–8.

[7] O. Georgiou and et al., "Low power wide area network analysis: Can lora scale?" *WCL*, 2017.

[8] Z. Xu, P. Xie, and J. Wang, "Pyramid: Real-time lora collision decoding with peak tracking," in *INFOCOM'21*.

[9] X. Xia and et al., "Ftrack: Parallel decoding for lora transmissions," *IEEE/ACM Transactions on Networking*, 2020.

[10] M. O. Shahid and et al., "Concurrent interference cancellation: Decoding multi-packet collisions in lora," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*.

[11] X. Wang and et al., "mlora: A multi-packet reception protocol in lora networks," in *2019 IEEE 27th International Conference on Network Protocols (ICNP)*.

[12] S. Tong and et al., "Colora: Enabling multi-packet reception in lora," in *INFOCOM'20*, 2020.

[13] D. Yang and et al., "Coexistence of ieee802.15.4 based networks: A survey," in *IECON*, 2010.

[14] ——, "Wireless coexistence between ieee 802.11- and ieee 802.15.4-based networks: A survey," *IJDSN*, vol. 7, no. 1, p. 912152, 2011.

[15] https://www.i-scoop.eu/internet-of-things-guide/iot-network-lora-lorawan/.

[16] S. Fahmida and et al., "Handling coexistence of lora with other networks through embedded reinforcement learning," in *IoTDI*, 2023, p. 410–423.

[17] Z. Liu and I. Elhanany, "RL-MAC: A QoS-aware reinforcement learning based MAC protocol for wireless sensor networks," in *ICNSC*, 2006, pp. 768–773.

[18] X. Huang and et al., "A reinforcement learning based medium access control method for lora networks," in *ICNSC*, 2020.

[19] "Ns-3," https://www.nsnam.org/.

[20] X. Xia, Y. Zheng, and T. Gu, "Ftrack: Parallel decoding for lora transmissions," in *SenSys*, 2019, pp. 192–204.

[21] A. Dongare and et al., "Charm: exploiting geographical diversity through coherent combining in low-power wide-area networks," in *IPSN*.

[22] A. Gadre and et al., "Frequency configuration for low-power wide-area networks in a heartbeat." in *NSDI*, 2020, pp. 339–352.

[23] G. Chen and et al., "Lofi: Enabling 2.4 ghz lora and wifi coexistence by detecting extremely weak signals," in *INFOCOM 2021*.

[24] K. L. A. Yau and et al., "Enhancing network performance in distributed cognitive radio networks using single-agent and multi-agent reinforcement learning," in *LCN*, 2010, pp. 152–159.

[25] R. Arroyo-Valles and et al., "Q-probabilistic routing in wireless sensor networks," in *2007 ISSNIP*, 2007, pp. 1–6.

[26] F. R. Yu and et al., "A new qos provisioning method for adaptive multimedia in wireless networks," *TVT*, vol. 57, no. 3, pp. 1899–1909, 2008.

[27] Z. Lu and et al., "Dynamic channel access and power control in wireless interference networks via multi-agent deep reinforcement learning," *TVT*.

[28] K.-L. A. Yau and et al., "Review: Reinforcement learning for context awareness and intelligence in wireless networks: Review, new features and open issues," *J. Netw. Comput. Appl.*, vol. 35, no. 1, pp. 253–267, Jan. 2012.

[29] H. Zhang and et al., "Reinforcement learning-based interference control for ultra-dense small cells," in *GLOBECOM'18*.

[30] F. Meyer and V. Turau, "Qma: A resource-efficient, q-learning-based multiple access scheme for the iiot," in *2021 ICDCS*.

[31] Y. Yu and et al., "Multi-agent q-learning algorithm for dynamic power and rate allocation in lora networks," in *2020 PIMRC*.

[32] M. A. Haque and et al., "A game-theoretic approach for mitigating jamming attacks in lpwan," *EWSN*, 2023.

[33] M. A. Haque and A. Saifullah, "Handling jamming in lora," *IoTDI*, 2024.

[34] L. alliance, "LoRaWAN specification," https://lora-alliance.org/resource-hub/lorawanr-specification-v11, 2017.

[35] A. Paszke and et al., "Pytorch: An imperative style, high-performance deep learning library," in *NeurIPS*.

[36] D. Magrin and et al., "Performance evaluation of lora networks in a smart city scenario," in *ICC (2017)*.

[37] H. Yin and et al., "Ns3-ai: Fostering artificial intelligence algorithms for networking research," in *Proceedings of the 2020 Workshop on Ns-3*, ser. WNS3 2020. New York, NY, USA: Association for Computing Machinery, 2020, p. 57–64. [Online]. Available: https://doi.org/10.1145/3389400.3389404

[38] "Ettus research," https://www.ettus.com/product/.

[39] "GNU Radio," http://gnuradio.org.

[40] "Dragino gps/lora shield," https://www.dragino.com/products/lora/item/102-lora-shield.html.

[41] "Arduino uno rev3," https://store-usa.arduino.cc/products/arduino-uno-rev3.