Adaptive Mobility Control in LoRaWAN via Reinforcement Learning

August F. Valentin, Simon K. Janum, F. Fernando Jurado-Lasso, Xenofon Fafoutis, Charalampos Orfanidis
Department of Applied Mathematics and Computer Science, Technical University of Denmark
{s194802, s194609}@student.dtu.dk, {ffjla, xefa, chaorf}@dtu.dk

***Abstract*—Low-Power Wide Area Networks (LPWANs), such as LoRaWAN, are pivotal for large-scale IoT deployments. However, traditional stationary gateways (GWs) impose scalability and cost constraints. We propose an AI-driven mobile GW architecture that leverages reinforcement learning (RL) to dynamically adapt GW mobility based on real-time network conditions. Our custom mobility module, integrated into OMNeT++ with FLoRa and TensorFlow Lite Micro, enables on-the-fly decision-making to optimize Packet Delivery Ratio (PDR) and fairness. The results of the simulations show that the introduced RL-GW achieves up to 99.94% PDR in high-fidelity simulations, outperforming static and heuristic mobile strategies. The RL policy generalizes robustly across network sizes and scenarios, offering a scalable, low-overhead solution for adaptive LPWAN infrastructure.**

***Index Terms*—IoT, LoRa, Scalability, Reinforcement Learning**

I. Introduction

Low-Power Wide Area Network (LPWAN) [1] are increasingly critical for Internet of Things (IoT) applications, enabling large-scale deployments in smart cities, agriculture, and environmental monitoring. Long Range (LoRa) [2], a leading LPWAN technology, has been widely studied for its scalability and coverage, but challenges remain in supporting large deployments efficiently [3], [4]. Long Range Wide Area Network (LoRaWAN) [5] is the Media Access Control (MAC) layer protocol built on top of LoRa technology. Traditional approaches to scaling LoRaWAN involve deploying additional gateways (GWs) [6], using the Adaptive Data Rate (ADR) mechanism to dynamically adjust the Radio Frequency (RF) settings [3], or by channel and frequency management to improve network capacity by utilizing multiple frequencies and non-overlapping channels [7]. However, these approaches have inherent limitations. The deployment of additional GW increases infrastructure costs and complexity [6], while ADR and frequency management offer only partial improvements under dynamic network conditions. A remaining challenge is scaling LoRaWAN deployments while maintaining reliable coverage and network performance. A promising alternative to scale LoRaWAN networks is to use mobile GWs that dynamically adapt to network demands. For instance, a GW integrated on an Unmanned Aerial Vehicle (UAV) could serve several end nodes. However, mobility introduces new challenges, including optimal path planning and real-time adaptation to varying network conditions. These challenges are inherently complex due to the stochastic nature of IoT deployments, where nodes exhibit diverse spatial distributions and transmission patterns.

This paper proposes an Reinforcement Learning (RL)-driven mobile GW architecture that dynamically adapts the mobility of the GWs based on real-time network conditions. The purpose is to explore the feasibility of using RL to enhance LoRaWAN scalability through mobile GWs. RL [8] presents a potential solution to optimize the behavior of the mobile GWs in real time, as it excels in decision making under uncertainty and dynamically adapts to dynamic network conditions, unlike supervised learning methods [9], which require predefined datasets. The proposed system employs an RL-based policy to govern GW mobility, with the objective of optimizing coverage and fairness. The fairness metric is the Jain's Fairness Index [10], which quantifies how evenly the Packet Delivery Ratio is distributed across all nodes, with a value of 1 indicating perfectly equal performance. This approach is evaluated against traditional static GW deployments and heuristic mobility strategies.

The evaluation relies on OMNeT++ [11], a high-fidelity network simulator, along with a custom lightweight RL environment built using Stable-Baselines3 (SB3) [12] to streamline training and testing. FLoRa [13], a framework built on OMNeT++ to simulate LoRa communication, serves as the simulation foundation. Since FLoRa lacks native support for AI-driven mobility, TensorFlow Lite Micro [14] is integrated with OMNeT++ to enable real-time inference within simulations. This setup supports RL model training in isolation, followed by seamless deployment into a simulated LoRaWAN scenario.

The remainder of this paper is organized as follows: Section II presents background on LoRaWAN and RL. Section III reviews the state of the art. Section IV illustrates the design and the implementation of the proposed solution. Section V validates the system and Section VI presents the evaluation across different scenarios. Section VII quantifies the impact of the model and Section VIII concludes the paper and discusses directions for future work.

II. Background

This section outlines the key technologies used in this paper around LoRa for communication and RL which enables adaptive mobility optimization.

LoRa [15] is a wireless communication technology designed for long-range, low-power communication. It utilizes Chirp

979-8-3503-9281-4/25/$31.00 ©2025 IEEE

Spread Spectrum (CSS) modulation, enhancing resilience against interference, making it suitable for mobile applications. LoRa operates in sub-GHz license-free bands (e.g., 868 MHz in Europe) and allows configurable parameters such as Bandwidth (BW), Spreading Factor (SF), and Transmission Power (TP) to regulate data rate, range, and energy efficiency.

LoRaWAN [16] is a MAC layer protocol that enables networked communication for LoRa nodes. It follows a star-of-stars topology where end nodes communicate with GWs, which forward messages to a network server. LoRaWAN employs ADR for dynamic parameter adjustment and enforces duty cycle constraints to mitigate network congestion. End devices are categorized into Class A (energy-efficient), Class B (scheduled downlink slots), and Class C (continuous reception). These properties make LoRaWAN well-suited for large-scale IoT deployments but introduce challenges in scalability, which this paper addresses using mobile GWs.

Framework for LoRa (FLoRa) [13] extends OMNeT++ [11] and INET by incorporating specific LoRaWAN modules, including end devices, GWs, and network servers. It allows fine-grained simulations of LoRaWAN deployments, supporting node mobility and adaptive configuration strategies. However, FLoRa does not support AI-driven mobility, necessitating modifications to integrate RL-based decision-making.

RL [8] is a decision-making framework where an agent interacts with an environment to learn an optimal policy by maximizing cumulative rewards. Unlike supervised learning [9], RL does not rely on labeled data but optimizes sequential decisions through trial and error [17], [18]. An RL problem is typically formulated as a Markov Decision Process (MDP), defined by:

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma) \tag{1}$$

where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P(s'|s,a)$ is the state transition probability, $R(s,a)$ is the reward function, and $\gamma \in [0,1]$ is the discount factor that balances immediate and future rewards. The objective is to learn an optimal policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes the expected return:

$$J(\pi) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_t\right]. \tag{2}$$

The agent interacts with the environment in a sequence: at time t, it observes state s_t, selects an action a_t, receives a reward r_t, and transitions to state s_{t+1}. Learning involves balancing exploration (discovering new strategies) and exploitation (using known strategies for rewards). This trade-off is managed through strategies like ϵ-greedy.

III. Related Work

The deployment and optimization of LoRaWAN GWs have been extensively studied in the literature, primarily focusing on static and mobile GW solutions. Machine Learning (ML)-based approaches have emerged, though they remain limited in application. This section reviews the most relevant works in static and mobile GW deployment, highlighting their contributions, limitations, and how they compare to our approach.

A. Static Gateway Deployment

Several studies focus on optimizing the placement of static LoRaWAN GWs to maximize network coverage, reduce energy consumption, and minimize interference. Pagano et al. [19] leverage graph theory and graph signal processing to optimize GW placement in water distribution networks, reducing the number of required GWs while maintaining connectivity. Loh et al. [20] propose a graph-based approach to minimize GW count while mitigating packet collisions. Mhatre et al. [21] introduce two algorithms—one based on connected component analysis and another using simulated annealing—to enhance network reliability and energy efficiency. Wixted et al. [22] evaluates LoRa through real-world experiments, showing that deploying multiple GWs significantly improves coverage, particularly in challenging terrain. Griva et al. [23] simulates LoRa deployments in open-field cultivation using FLoRa, demonstrating that increasing the number of GWs from 1 to 4 improves network performance by 28.6%, while increasing the number of nodes from 10 to 1000 raises the collision rate from 1.7% to 65.7%.

Other works focus on optimizing network parameters alongside GW placement. Loubany et al. [24] improve energy efficiency by optimizing SF selection and transmission power. Correia et al. [25] compare clustering algorithms to optimize GW deployment in smart agriculture. Similarly, Matni et al. [26] develop DPLACE, a clustering-based model for dynamic IoT GW placement. Ousat et al. [27] formulate a mixed-integer non-linear optimization problem to jointly optimize GW placement and power allocation, proposing an approximate algorithm for large-scale networks.

While these approaches significantly improve GW placement strategies, they primarily assume fixed network topologies and do not consider adaptive repositioning based on real-time network conditions. Moreover, none of these works incorporate AI-driven mechanisms to dynamically adjust GW placement in response to network variations. Our work distinguishes itself by integrating ML to enable adaptive decision-making, ensuring efficient and scalable LoRaWAN deployments.

B. Mobile Gateway Deployment

To improve network adaptability, several studies propose mobile GW solutions, leveraging Unmanned Aerial Vehicles (UAVs), vehicles, and other mobile platforms for dynamic data collection and coverage extension. Islam et al. [28] utilize ML to estimate the distance between nodes and a mobile GW, enabling more accurate localization. Chen et al. [29] introduce LoRaDrone, a UAV-based mobile GW system designed to optimize energy consumption through dynamic channel allocation. Stellin et al. [30] propose LoRaUAV, a two-layer network where UAVs relay traffic between mobile LoRaWAN nodes and a base station, enhancing connectivity in post-disaster scenarios. Other works explore domain-specific applications. Soares et al. [31] design a mobile LoRa GW for railway networks, using control messages for coordinated communication. Ikhsan et al. [32] analyze the performance of mobile and static GWs in livestock monitoring, demonstrating the cost benefits

of mobile solutions in large areas. Sobhi et al. [33] investigate vehicle-mounted mobile GWs for IoT data collection, analyzing trade-offs between data freshness, throughput, and energy consumption. Additional studies, such as [34] and [35], examine mobile LoRa GWs for flood monitoring and smart electricity metering, respectively. Gutiérrez et al. [36] propose mobile LoRa GWs for agriculture, improving data collection from environmental sensors.

Works surveyed here primarily focus on specific applications and do not address broader scalability and reliability challenges in large-scale networks. Furthermore, most of these works rely on predefined mobility patterns rather than AI-based optimization techniques to dynamically adjust GW movement based on real-time network conditions. In contrast, our approach integrates RL and transfer learning to enable intelligent, adaptive decision-making, optimizing both GW mobility and network performance.

Existing research confirms that multiple GWs significantly enhance network coverage, reliability, and energy efficiency. Real-world experiments [22] demonstrate that deploying multiple GWs mitigates terrain-induced connectivity issues, while simulations [23] show that increasing the number of GWs improves performance. However, static GW deployment remains costly and lacks adaptability to dynamic network conditions. Mobile GWs provide a cost-effective alternative, yet existing approaches often rely on predefined mobility patterns rather than AI-based optimization. While UAV- and vehicle-mounted GWs [29], [33] improve coverage and energy efficiency, they do not dynamically adapt to real-time network variations. Our work differentiates itself by integrating RL and transfer learning to enable intelligent, adaptive GW movement. Unlike prior studies, our approach continuously optimizes deployment based on real-time network conditions, enhancing scalability, efficiency, and resilience in LoRaWAN deployments.

IV. Design and Implementation

The system we introduce was designed based on OMNeT++ and SB3. The latter was used for model training in a simplified environment, where the trained model was then transferred to OMNeT++ for inference. Although experiments were conducted with training directly in OMNeT++, this approach proved too time-consuming to be practical. Therefore, the training process was conducted in SB3, where mobility, LoRa, and lower-level components were abstracted or simplified to speed up training while preserving essential decision-making elements. Fig. 1 presents an overview of the system.

A. Custom Environment in Stable-Baselines3

The purpose here was to accelerate the training process while allowing for easier and more flexible modifications and implementations. To support development, a rendering of the environment was created to monitor components during implementation and observing the agent's behavior after training. The rendering is shown in Fig. 2. The movement area was visualized as a red box, the RL-GW (agent), was visualized as a white rectangle, and four nodes were visualized with green circles. The time was represented as discrete *steps*, where each step incremented the time by one unit. Since the agent moved at a fixed speed (expressed in meters per step, equivalent to m/s), this temporal progression could also be interpreted as seconds. However, for consistency we refer to this progression as steps.

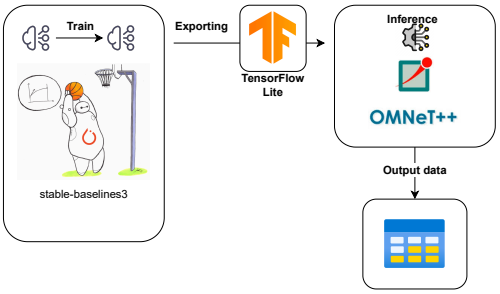


Fig. 1: System design for training and evaluation.

Each node was characterized by a set of key attributes: *position*, *time of first packet*, *transmission interval*, and *transmission model*. The *time of first packet* specified the step at which the node transmitted its first packet. For instance, a value of 150 indicates that the first packet is transmitted at step 150. The *transmission interval* defined the frequency at which a node transmitted packets, measured in discrete steps. Furthermore, *transmission spacing* denotes the time interval (in steps) between two consecutive transmitted packets.

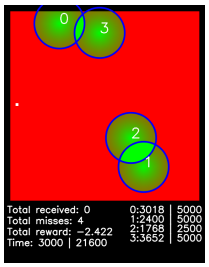


Fig. 2: RL-GW with a small white rectangle serving 4 nodes in SB3.

Once the first packet was transmitted, subsequent transmission times were calculated based on the transmission interval, incorporating a small degree of random variation. This randomness was introduced through a truncated normalization function, which takes the transmission interval as input with a standard deviation of 5 steps. The intention behind this variation was to avoid strictly fixed transmission intervals, thereby creating a more dynamic and realistic representation of packet transmission behavior.

To maintain variability across episodes (each episode is an independent $21,600$-second simulation) and reduce the risk of overfitting to specific configurations, the environment was re-initialized with randomized node and GW positions, as well as randomized transmission intervals. This dynamic environment encouraged the model to generalize its learning, focusing on efficiently estimating packet times and effectively navigating between nodes.

Each node was associated with a *transmission model* that determined the success of packet reception. While transmission could be successful, reception depended on two main factors. First, packets were always lost if the GW was outside

the node's transmission range. Second, within range, the probability of reception decreased with distance from the transmitting node, making distant GWs more prone to packet loss. This model balances simplicity and realism by capturing key behaviors without excessive complexity. The probability of successful reception is given by the following equation.

$$P(\textit{Packet Reception}) = e^{-\frac{\textit{distance}}{\textit{ploss scale}}} \tag{3}$$

Here, the parameter of the *ploss scale* regulated the probability decay, effectively controlling the influence of distance on reception success. During the evaluation, the *ploss scale* was set to 50, with a max transmission radius of what corresponds to 400 meters per node. Consequently, when the GW was on the edge of the transmission radius, the probability of successful packet reception was approximately 44.9%. The chosen value of the *ploss scale* resulted in an aggressive decay of the reception probability with increasing distance. This aggressive configuration reinforced the importance of proximity during learning, making the trade-off between distance and reception success more apparent to the agent. While simplified, the hard cutoff reflects practical LoRa constraints and provided a clear boundary for decision-making.

The *policy* defines how the RL-GW selects movement actions based on observed network conditions. The observation space was carefully designed and is having a direct impact on the decision-making. The policy utilizes a fully-connected Artificial Neural Network (ANN). The *default PPO policy network architecture* from SB3 was employed, with a slight adjustment to improve learning efficiency. Specifically, the network consisted of three fully connected layers for both the *actor* and *critic* components, and the intermediate layers used *tanh activation functions*. This configuration was found to accelerate convergence compared to shallower architectures. PPO was selected due to its effectiveness in environments requiring reward exploration and strategic behavior, where it generally outperforms value-based methods such as DQN, albeit with increased sensitivity to hyper-parameters [37].

The *observation space* was defined as $obs(\Delta t, \theta, \Delta d)$. Δt is the expected time until each node transmits a packet. θ stands for the direction of the GW to each node (in degrees). Δd is the distance between the GW and each node. In an environment with 4 nodes, the observation space comprises 12 values, as each of the three input components is calculated per node. To ensure consistent performance across varied conditions, the observations were normalized. Namely, *distance*, (Δd) was normalized by dividing by the maximum distance of the environment. θ was normalized by dividing by 360. *Expected time*, (Δt) was normalized by dividing by the maximum expected time interval. Normalization was crucial to balance the scale of different inputs, preventing dominance by larger values. This also promoted uniformity, making the agent's learning process more stable and efficient.

The agent had a set of 5 *discrete actions*: *Up, Down, Left, Right, and Standing still*. The action was chosen using *softmax function*. We used a discrete action space to reduce training complexity and ensure deterministic movement control in OMNeT++, which simplified integration and produced stable convergence. During training, it was observed that increasing the impact of each action led to improved learning dynamics. To enhance this effect, the agent was configured to take one action every 10 simulation steps rather than at each step. This adjustment made each decision more consequential, reinforcing the learning of strategic movement.

An effective credit assignment strategy is essential for the RL-based mobility model to optimize GW movement and packet reception. The reward structure was designed to encourage the agent to maximize packet delivery. The primary reward components are the *positive reception reward* for correctly receiving a packet the *missed packet penalty* which is a negative reward, scaled by the distance from the transmitting node to reduce the penalty when packet loss occurs despite proximity, the *positional reward*, a small positive reward for reducing the distance to the next transmitting node and the *directional reward* a minor positive reward for moving in the direction of the next transmitting node. This credit assignment structure encourages the agent to maintain proximity to the transmitting nodes while minimizing unnecessary movements. The use of distance-based scaling for missed packet penalties ensures that the model prioritizes proximity-based decision-making.

The model was trained using the PPO algorithm for a total of 10 million steps, where each step corresponds to a single action. Due to a frame skip of 10, each action is applied across 10 steps, resulting in 100 million base environment steps. This reduces the frequency of decision making, thus increasing the impact of each action and improving learning efficiency.

Each episode spans $21\,600\,\mathrm{s}$, with a simulation time step of $1\,\mathrm{s}$, yielding 2160 actions per episode. The Neural Network (NN) policy uses an extended PPO architecture with four fully connected layers for both the actor and critic networks. The actor network contained 226 nodes and 9280 edges, while the critic network includes 218 nodes and 9024 edges. In total, the model consisted of 8 layers, 444 nodes, and $18,304$ weights. Only the actor network is required at inference time, making the model compact and suitable resource limited IoT systems.

B. Integrating OMNeT++ for Model Evaluation

To evaluate models trained in the simplified environment of SB3, we employ OMNeT++ [11] and the FLoRa framework [13], to accurately model the unique physical signal characteristics of LoRa technology. This approach allows to transition from trained models and low-fidelity environment to a high-fidelity simulation, enabling robust performance assessment under more realistic conditions. Implementing the simulation required the integration and cooperation of multiple frameworks within OMNeT++, including INET, FLoRa and TF-lite-micro.

INET provides mobility models, such as circular or linear (tractor-like) motion. However, these models are static, with all movement defined at initialization and no support for runtime updates. While INET's "turtle" module allows scripted

trajectories via XML, it similarly lacks dynamic decision-making. To enable policy-driven mobility, a custom module was developed, inheriting from `INET_MOBILITY_BASE`. It integrates with a dedicated inference component, which executes NN inference at regular intervals. Based on the output, the GW updates its direction and continues the motion until the next inference step.

V. System Validation

To ensure that RL models trained in SB3 generalize robustly to OMNeT++, we validate by comparing performance metrics in both environments. The purpose is to assess whether SB3 environment sufficiently approximates the OMNeT++ simulation, making it a viable training framework.

A. Experimental Setup

We consider a controlled scenario where a RL-GW operates in a LoRaWAN network with four static nodes. The same network topology is implemented in both SB3 and OMNeT++, with differences in scale due to the representation of distances in each environment. Both simulations are performed for 100 episodes and a different random seed generated in each episode. The parameters are listed in Table I for both SB3 and OMNeT++ respectively. Note that truncnorm is an abbreviation of normal distributed, truncated to positive values, and the x and y coordinates are an order of magnitude larger for the OMNeT++ case.

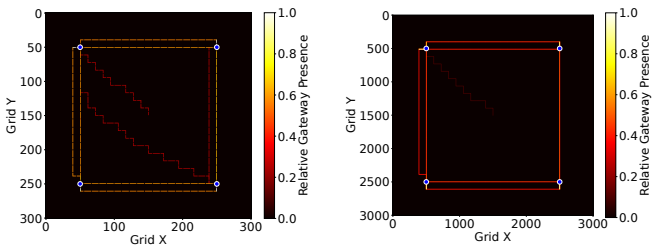
Node ID	Position (x,y) [m]	First Packet Time [s]	Transmission Interval [s]
Node 0	(50,50)	400	`truncnorm(1600,5)`
Node 1	(50,250)	800	`truncnorm(1600,5)`
Node 2	(250,250)	1200	`truncnorm(1600,5)`
Node 3	(250,50)	1600	`truncnorm(1600,5)`
GW	(150,150)	N/A	N/A

TABLE I: Network topology parameters.

To ensure that the RL model was correctly exported from SB3 to TensorFlow and converted to TFLite, a consistency check was performed by generating random input samples and comparing the output probabilities of both models. The input dimensions were consistent and the outputs were compared using the `numberpy.allclose` function. The validation showed minimal differences (Mean Absolute Difference: $7.51\,10^{-8}$ and Mean Relative Difference: $1.39\,10^{-6}$) likely due to numerical precision variations.

Once integrated into OMNeT++, the behavior of the model was analyzed to ensure consistency with SB3. OMNeT++ demonstrated a PDR of 99.94%, compared to 90.32% for SB3. Furthermore, the fairness metric was 1.0 for OMNeT++ and 0.99 for SB3, confirming that the RL-GW served the nodes in a fair and balanced way.

The RL-GW's movement pattern is cyclic, reducing its distance to each node before transmissions. Fig. 3 illustrates the mobility pattern for both environments. To visualize the mobility behavior, a post-processing pipeline generated a heatmap from logged positions. These positions were mapped onto a high-resolution grid and interpolated to reconstruct the traversal path. A Gaussian filter smoothed the paths, and logarithmic scaling produced an intensity map, highlighting areas of frequent presence.



(a) (b)

Fig. 3: Mobility of the RL-GW during training with SB3 (a) and for inference with OMNeT++ (b) for a single episode.

The results confirm the successful model transfer from SB3 to OMNeT++, demonstrating its generalization. Furthermore, they suggest that SB3 provides a conservative approximation of more realistic conditions for robust training conditions.

VI. Evaluation

This section compares the performance of an RL-GW to stationary GWs in a LoRaWAN network across various scenarios. It starts with a single mobile GW versus four stationary, each tested in 100 episodes with different random seeds. Then the evaluation explores static versus RL-based mobility and assesses scalability, ultimately comparing results to state-of-the-art solutions.

The first scenario compares the performance of an RL-GW to four stationary GWs in OMNeT++. The goal is to evaluate how effectively a single RL-GW can replace multiple fixed ones, particularly under constrained conditions. While previous results in Section V showed that the RL-GW can serve four end-nodes with near-perfect performance, this scenario introduces a more challenging setup by reducing the transmission interval. Table II lists the parameters for this scenario. The increased traffic requires the RL-GW to make consistently optimal decisions, testing its adaptability. The stationary GWs are placed near each node (100 meters along the x-axis), representing an ideal fixed deployment.

This setup establishes a baseline for subsequent comparisons, demonstrating that an RL-GW can match the coverage of a four GW static deployment under specific conditions. To determine appropriate transmission intervals, the required travel time between sequentially transmitting nodes was first estimated. In a grid layout with 2000 m spacing, and assuming adjacent nodes transmit in sequence, the RL-GW moving at $11\,m/s$—requires approximately $2000/11 \approx 181$ s to traverse the distance. This value represents the lower bound for transmission spacing to ensure reliable reception. However, ideal conditions are unlikely: the RL-GW can only update its movement direction every 10 s due to the inference rate of the policy, potentially causing overshooting and the need

Node ID	Position (x,y) [m]	First Packet Time [s]	Transmission Interval [s]
Node 0	(500,500)	400	`truncnorm(1000,5)`
Node 1	(500,2500)	800	`truncnorm(1000,5)`
Node 2	(2500,2500)	1200	`truncnorm(1000,5)`
Node 3	(2500,500)	1600	`truncnorm(1000,5)`
RL GW	(1500, 1500)	N/A	N/A
St. GW 0	(400, 500)	N/A	N/A
St. GW 1	(400, 2500)	N/A	N/A
St. GW 2	(2600, 2500)	N/A	N/A
St. GW 3	(2600, 500)	N/A	N/A

TABLE II: OMNeT++ parameters for a RL-GW and four static GWs

for backtracking. Furthermore, the RL-GW relies on estimated transmission times, introducing additional uncertainty. To compensate for these limitations, the interval between transmissions was increased to 250 s. Consequently, each node is assigned a transmission interval of $n_{\text{nodes}} \cdot \Delta T_{\text{transmissions}} = 4 \cdot 250 = 1000\,\text{s}$.

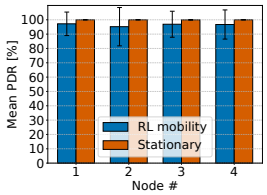


Fig. 4: PDR of static and RL-GWs.

Fig. 4 shows that the PDR of the stationary GWs is 100% as expected, since each stationary GW is allocated to serve only a single node, which is spatially adjacent. The fairness is (1.0) here as each node has 100 % throughput with the stationary GWs. The performance of the RL-GW is affected by the time constraints between transmissions. The RL-GW achieves an overall network PDR of 96.70%, and fairness of 0.99. Even though the performance is not completely consistent, is still remarkable, indicating that the model isn't entirely robust. This highlights that the time between transmissions is a critical aspect of the RL-GW's performance.

In LoRaWAN the GW's mobility plays a critical role in network performance. Deterministic mobility, which follows a predetermined trajectory, contrasts with dynamic mobility driven by RL, where the GW adapts its movement to network conditions. This scenario evaluates the performance of both approaches under various conditions.

We begin by comparing a deterministic mobility GW moving in a circular pattern with an RL-GW. For the deterministic mobility GW, the motion is defined by a set of parameters such as center position, radius, linear velocity, and initial angular position. Conversely, the RL-GW adapts its movement dynamically to optimize performance based on real-time conditions. Initially we investigate the performance of both the RL and deterministic mobility GWs when the end nodes are positioned equidistantly on a circle. The nodes transmit at harmonic frequencies with equal time offsets. In this case, the deterministic mobility GW follows a circular trajectory with an angular velocity of $\omega = \frac{\pi}{500}\ \frac{\text{rad}}{\text{s}}$. The radius of the motion is 1000 meters, and the linear velocity is approximately 6.28 m/s. The initial angular position of the GW is calculated to ensure that it reaches the first node at its transmission time. Table III provides the OMNeT++ configuration for this case.

Node ID	Position (x, y) [m]	First Packet Time [s]	Transmission Interval [s]
Node 0	(1500, 1500)	400	`truncnorm(1000, 5)`
Node 1	(2500, 500)	650	`truncnorm(1000, 5)`
Node 2	(3500, 1500)	900	`truncnorm(2000, 5)`
Node 3	(2500, 2500)	1150	`truncnorm(2000, 5)`
RL GW	(2500, 1500)	N/A	N/A

TABLE III: OMNeT++ configuration while the nodes are evenly positioned on a circular mobility pattern.

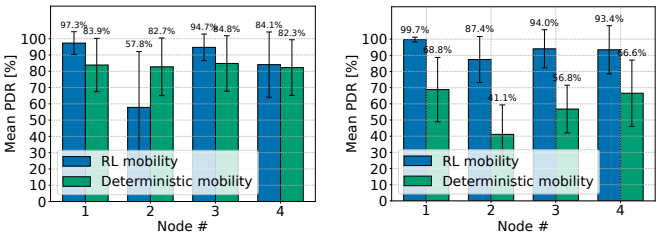
Fig. 5 (a) depicts the PDR, and although the deterministic mobility GW is configured specifically for this scenario, the mean PDR per node is less than 85% of all nodes. This must be due to the stochastic deviation in transmission times, which the deterministic mobility is not able to adjust to. The RL-GW, though comparable in average performance, shows a larger variance in PDR, reflecting its sensitivity to real-time conditions. The total PDR is 83.51% for RL-GW and 83.42% for the deterministic mobility. Although the overall PDR of the network looks similar, the fairness is 0.92 for the RL-GW and 0.97 for the deterministic mobility due to the fact that the latter cannot serve Node 2 because it is further away compared to the other nodes.

Node ID	Position (x, y) [m]	First Packet Time [s]	Transmission Interval [s]
Node 0	(1700, 1400)	400	`truncnorm(1000, 5)`
Node 1	(2700, 800)	650	`truncnorm(1000, 5)`
Node 2	(3800, 1450)	900	`truncnorm(2000, 5)`
Node 3	(2400, 2300)	1150	`truncnorm(2000, 5)`
RL GW	(2500, 1500)	N/A	N/A

TABLE IV: OMNeT++ configuration while the nodes are dispersed related to a circle.

Next we define a more realistic scenario where nodes are not perfectly arranged on a circle where the nodes are shifted slightly from the circular mobility GW's path. The configuration parameters are described in Table IV. To ensure an impartial comparison, the circular motion of the static GW is adjusted so that its linear velocity matches that of the RL GW ($\approx$ 11 m/s). The center of the motion is placed at the midpoint of the nodes, and the radius is calculated so that the time to complete a full rotation aligns with the nodes' transmission interval. The radius of the circular motion is approximately 1751 meters, and the motion intersects the grid at two points per side, near the corners. Fig. 5 (b) shows the results. The deterministic mobility GW's performance degrades significantly due to the deviation from the ideal circular arrangement. Specifically, Node 2 experiences very low PDR, as its position deviates the most from the static GW's mobility pattern. The RL-GW performs better, as it adapts more effectively to the varied node positions. This demonstrates the flexibility of the RL approach in handling non-ideal conditions. The total network PDR is 93.65% for

RL-GW and 58.32% for deterministic mobility, highlighting the inability to serve the nodes in that case. The fairness is 0.99 and 0.90 for RL-GW and deterministic mobility respectively.



(a) (b)

Fig. 5: PDR for RL and deterministic mobility with evenly (a) and randomly (b), deployment on a circular mobility pattern.

Real-world LoRaWAN deployments may consist of hundreds or thousands of devices. Therefore, a new large-scale scenario was designed to evaluate this approach. Each episode contained 20 end nodes randomly placed in a $3000\,\mathrm{m} \times 3000\,\mathrm{m}$ area. Transmissions were spaced by 175 s, resulting in a per-node interval of 3500 s. The extended interval prevents overlapping transmissions and removes the possibility of collisions. To establish a benchmark, the performance of the RL-GW was compared to a fixed deployment using four statically placed GWs. These were deployed based on a heuristic method [21], clustering nodes into four regions based on communication range. The policy architecture was adapted to handle an arbitrary number of nodes by dynamically selecting the four most imminent transmitters based on expected transmission times. This allows the agent to generalize its behavior without modifying the NN structure.

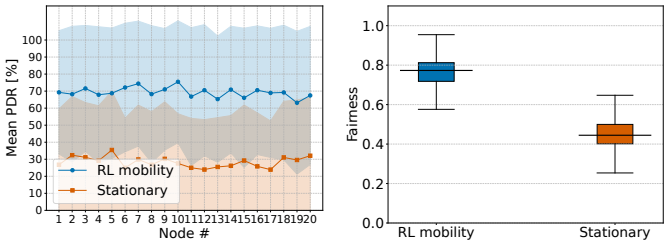
Fig. 6 summarize the results. Despite the increased number of nodes and reduced movement time between events, the RL-GW achieves an overall PDR of 76% and average fairness of 0.76. These results are noteworthy, considering the policy was trained exclusively on 4-node episodes. In contrast, the heuristic method achieves only 28.35% PDR and fairness of 0.45. While both methods exhibit variability across episodes, the RL-GW performs more consistent.

These findings confirm that the proposed RL-GW scales beyond the training configuration and significantly outperforms conventional static placement methods under non-overlapping transmission conditions.

VII. Model Complexity and Scalability

To assess the practicality and scalability of the proposed system on real-world embedded platforms, we outline its computational footprint and generalization to larger networks.

The trained model is exported in the TFLite format, with a size of 40 kB. Inference involves 226 nodes and 9280 weights. To estimate its execution time and energy consumption on embedded hardware, prior results from [38] are used: a 410 kB model deployed on a `SparkFun Edge` microcontroller achieved inference in 2000 ms. Assuming linear



(a) (b)

Fig. 6: PDR (a) and fairness (b) for a large-scale scenario.

scaling with model size, the 40 kB model is projected to complete inference in under 200 ms, consuming approximately $0.15\,\mu$Wh per step. Given a decision interval of 10 s, this corresponds to less than 2% CPU usage and negligible energy overhead, supporting feasibility for real-time, battery-powered deployment on platforms such as the Raspberry Pi 4 or low-power MCUs. Although the model was trained with 4 nodes, a sorting mechanism based on expected transmission times, allows the policy to generalize to arbitrary node counts by dynamically selecting the next 4 most relevant nodes. This enables scalability without modifying the NN architecture.

VIII. Conclusion

This paper presents a novel reinforcement learning (RL)-driven mobility framework for LoRaWAN GWs, offering a dynamic alternative to traditional static and heuristic-based mobile deployments. Through rigorous experimentation in both simplified and high-fidelity simulations, we demonstrate that RL-GW consistently achieves high PDR (up to 99.94 %) and fairness, even in challenging and large-scale scenarios. The system effectively generalizes beyond its training environment, adapting to diverse node placements and transmission schedules. Its compact policy model, lightweight inference footprint, and real-time adaptability make it highly suitable for resource-constrained IoT edge devices. Compared to static deployments and predefined mobility paths, the RL approach shows marked improvements in scalability, robustness, and responsiveness to real-time network conditions.

The introduced system operates under the assumption of known node locations and fixed transmission schedules. In more diverse and larger deployments, variations such as GPS inaccuracies or irregular transmission patterns could be addressed by using probabilistic models or enhanced localization infrastructure and methods [39]. Additionally, our model performs inference at 10 s intervals to optimize responsiveness and energy efficiency. These challenges could be tackled by using multi-agent coordination strategies to respond efficiently to dynamic network conditions. Therefore, we plan to extend this work by incorporating mobile end-device scenarios and urban multipath interference into the simulations, benchmarking against state-of-the-art solutions such as LoRaDrone [40], and developing multi-gateway coordination strategies to mitigate performance drops in dense networks. Future efforts will also

replace the simplified path loss model with empirical terrain-aware propagation models, conduct hardware validation of the TFLite model on edge devices to measure real-world latency and energy consumption, and clarify deployment constraints including GPS requirements and backhaul connectivity.

REFERENCES

[1] K. Mekki, E. Bajic, F. Chaxel, and F. Meyer, "A comparative study of lpwan technologies for large-scale iot deployment," *ICT Express*, vol. 5, no. 1, pp. 1–7, 2019.

[2] LoRa Alliance, "About lora alliance," 2025, accessed: 2025-04-17. [Online]. Available: https://lora-alliance.org/about-lora-alliance/

[3] K. Mikhaylov, J. Petäjäjärvi, and J. Janhunen, "On lorawan scalability: Empirical evaluation of susceptibility to inter-network interference," in *2017 European Conference on Networks and Communications (EuCNC)*, 2017, pp. 1–6.

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

[5] LoRa Alliance, "About LoRaWAN®," https://lora-alliance.org/about-lorawan/, 2025, accessed: 2025-03-27.

[6] M. C. Bor, U. Roedig, T. Voigt, and J. M. Alonso, "Do lora low-power wide-area networks scale?" in *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, ser. MSWiM '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 59–67. [Online]. Available: https://doi.org/10.1145/2988287.2989163

[7] B. Reynders, Q. Wang, P. Tuset-Peiro, X. Vilajosana, and S. Pollin, "Improving reliability and scalability of lorawans through lightweight scheduling," *IEEE Internet of Things Journal*, vol. 5, no. 3, 2018.

[8] A. K. Shakya, G. Pillai, and S. Chakrabarty, "Reinforcement learning algorithms: A brief survey," *Expert Systems with Applications*, 2023.

[9] P. Cunningham, M. Cord, and S. J. Delany, "Supervised learning," in *Machine learning techniques for multimedia: case studies on organization and retrieval*. Springer, 2008, pp. 21–49.

[10] U. U. Khan, N. Dilshad, M. H. Rehmani, and T. Umer, "Fairness in cognitive radio networks: Models, measurement methods, applications, and future research directions," *Journal of Network and Computer Applications*, vol. 73, pp. 12–26, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804516301527

[11] A. Varga, "Omnet++," in *Modeling and tools for network simulation*. Springer, 2010, pp. 35–59.

[12] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html

[13] M. Slabicki, G. Premsankar, and M. Di Francesco, "Adaptive configuration of lora networks for dense iot deployments," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, 04 2018.

[14] T. Authors, "Tensorflow lite for microcontrollers," https://www.tensorflow.org/lite/microcontrollers, 2025, accessed: 2025-04-04.

[15] M. Bor, J. Vidler, and U. Roedig, "Lora for the internet of things," in *Proceedings of the 2016 International Conference on Embedded Wireless Systems and Networks*, ser. EWSN '16. USA: Junction Publishing, 2016, p. 361–366.

[16] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, "Understanding the limits of lorawan," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 34–40, 2017.

[17] M. A. Wiering and M. Van Otterlo, "Reinforcement learning," *Adaptation, learning, and optimization*, vol. 12, no. 3, p. 729, 2012.

[18] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE communications surveys & tutorials*, vol. 21, no. 4, 2019.

[19] A. Pagano, D. Garlisi, F. Giuliano, T. Cattai, and F. Cuomo, "Application-Aware LoRaWAN Gateway Placement in Massive IoT Water Distribution Networks," in *2024 IEEE 10th World Forum on Internet of Things (WF-IoT)*, Nov. 2024, pp. 475–480.

[20] F. Loh, N. Mehling, S. Geißler, and T. Hoßfeld, "Efficient graph-based gateway placement for large-scale LoRaWAN deployments," *Computer Communications*, vol. 204, pp. 11–23, Apr. 2023.

[21] J. Mhatre, A. Lee, H. Lee, T. N. Nguyen, and K. Suo, "Improving Energy Efficiency and Reliability by Optimizing Gateway Placement in Dense LoRaWAN Networks," in *2024 IEEE 10th World Forum on Internet of Things (WF-IoT)*, Nov. 2024, pp. 340–345.

[22] A. J. Wixted, P. Kinnaird, H. Larijani, A. Tait, A. Ahmadinia, and N. Strachan, "Evaluation of LoRa and LoRaWAN for wireless sensor networks," in *2016 IEEE SENSORS*, Oct. 2016, pp. 1–3.

[23] A. Griva, A. D. Boursianis, S. Wan, P. Sarigiannidis, G. Karagiannidis, and S. K. Goudos, "Performance Evaluation of LoRa Networks in an Open Field Cultivation Scenario," in *2021 10th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, Jul. 2021.

[24] A. Loubany, S. Lahoud, A. E. Samhat, and M. El Helou, "Improving Energy Efficiency in LoRaWAN Networks with Multiple Gateways," *Sensors*, vol. 23, no. 11, p. 5315, Jan. 2023.

[25] F. P. Correia, S. R. da Silva, F. B. S. de Carvalho, M. S. de Alencar, K. D. R. Assis, and R. M. Bacurau, "LoRaWAN Gateway Placement in Smart Agriculture: An Analysis of Clustering Algorithms and Performance Metrics," *Energies*, vol. 16, no. 5, p. 2356, Jan. 2023.

[26] N. Matni, J. Moraes, H. Oliveira, D. Rosário, and E. Cerqueira, "LoRaWAN Gateway Placement Model for Dynamic Internet of Things Scenarios," *Sensors*, vol. 20, no. 15, p. 4336, Jan. 2020.

[27] B. Ousat and M. Ghaderi, "LoRa Network Planning: Gateway Placement and Device Configuration," in *2019 IEEE International Congress on Internet of Things (ICIOT)*, Jul. 2019, pp. 25–32.

[28] K. Z. Islam, D. Murray, D. Diepeveen, M. G. K. Jones, and F. Sohel, "LoRa localisation using single mobile gateway," *Computer Communications*, vol. 219, pp. 182–193, Apr. 2024.

[29] C. Chen, J. Luo, Z. Xu, R. Xiong, Z. Yin, J. Lin, and D. Shen, "LoRaDrone: Enabling Low-Power LoRa Data Transmission via a Mobile Approach," in *2022 18th International Conference on Mobility, Sensing and Networking (MSN)*, Dec. 2022, pp. 239–246.

[30] M. Stellin, S. Sabino, and A. Grilo, "LoRaWAN Networking in Mobile Scenarios Using a WiFi Mesh of UAV Gateways," *Electronics*, vol. 9, no. 4, p. 630, Apr. 2020.

[31] J. Soares, M. Luís, and S. Sargento, "Mobile LoRa Gateway for Communication and Sensing on the Railway," in *2023 IEEE Symposium on Computers and Communications (ISCC)*, Jul. 2023, pp. 499–502.

[32] M. G. Ikhsan, M. Y. A. Saputro, D. A. Arji, R. Harwahyu, and R. F. Sari, "Mobile LoRa Gateway for Smart Livestock Monitoring System," in *2018 IEEE International Conference on Internet of Things and Intelligence System (IOTAIS)*, Nov. 2018, pp. 46–51.

[33] S. Sobhi, A. Elzanaty, M. Y. Selim, A. M. Ghuniem, and M. F. Abdelkader, "Mobility of LoRaWAN Gateways for Efficient Environmental Monitoring in Pristine Sites," *Sensors*, vol. 23, no. 3, p. 1698, Jan. 2023.

[34] F. A. Saparudin, M. S. Halim, N. S. Amani Ibrahim, and M. F. Baharom, "Real-Time Flood Monitoring Using Mobile LoRaWAN IoT Gateway: A Case Study of the Muar River Catchment," in *2024 IEEE 1st International Conference on Communication Engineering and Emerging Technologies (ICoCET)*, Sep. 2024, pp. 1–4.

[35] S. Sugianto, A. A. Anhar, R. Harwahyu, and R. F. Sari, "Simulation of Mobile LoRa Gateway for Smart Electricity Meter," in *2018 5th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, Oct. 2018, pp. 292–297.

[36] S. Gutiérrez, I. Martínez, J. Varona, M. Cardona, and R. Espinosa, "Smart Mobile LoRa Agriculture System based on Internet of Things," in *2019 IEEE 39th Central America and Panama Convention (CONCAPAN XXXIX)*, Nov. 2019, pp. 1–6.

[37] N. D. L. Fuente and D. A. V. Guerra, "A comparative study of deep reinforcement learning models: Dqn vs ppo vs a2c," 2024. [Online]. Available: https://arxiv.org/abs/2407.14151

[38] P.-E. Novac, G. Boukli, A. Pegatoquet, B. Miramond, and V. Gripon, "Quantization and deployment of deep neural networks on microcontrollers," *Sensors (Basel, Switzerland)*, vol. 21, 04 2021.

[39] D.-H. Kim and J.-Y. Pyun, "Mobility identification utilizing deep learning for lorawan location-based services," *IEEE Wireless Communications Letters*, vol. 14, no. 1, pp. 193–197, 2025.

[40] C. Chen, J. Luo, Z. Xu, R. Xiong, Z. Yin, J. Lin, and D. Shen, "Loradrone: Enabling low-power lora data transmission via a mobile approach," in *2022 18th International Conference on Mobility, Sensing and Networking (MSN)*, 2022, pp. 239–246.